MICR'ORIC

LE MAGAZINE DES UTILISATEURS D'ORIC

.EN SAVOIR PLUS SUR LE D.O.S.
.DOMINEZ VOTRE CLAVIER
.LES COCKTAILS

.DES JEUX NOUVEAUX



N°6 – Automne 84 – 30 F.

Maintenan

La voici, votre imprimante.
Une véritable imprimante traceuse type Centronics, mode graphique ou alphanumérique, 4 couleurs (vert, rouge, noir et bleu), papier standard en bobine. Magnifique résolution, édition sur 40 ou 80 colonnes à la vitesse de 12 caractères/seconde. C'est l'esclave docile de votre ordinateur personnel. C'est elle que vous attendiez !... alors, allez-y, maintenant !

Le voici, votre ordinateur personnel.
L'ORIC ATMOS : 48K de mémoire, 8 couleurs à l'écran/mode graphique sur 200 x 240 pixels/clavier ergonomique professionnel de 57 touches/mode texte sur 28 lignes de 40 caractères ASCII, plus 80 caractères définissables, entrées et sorties pour extensions et périphériques...
Il s'adapte sur tous moniteurs ou téléviseurs grâce aux raccordements disponibles.
C'est lui que vous attendiez! ...alors, allez-y, maintenant !

POWER ON
PAPER END
LINE FEED
COLOUR SELECT
PEN CHANGE
ORIC

ATMOS de ORIC : l'ordinateur définitif.

t, allez-y !



La voici, votre mémoire de masse.
L'ORIC MICRO-DISC, il utilise les nouvelles disquettes de 3 pouces double face-double densité, sous carter de sécurité rigide. Capacité de 160K octets par face. Vitesse de débit 250Ko/s. Ces lecteurs sont extensibles jusqu'à 4 unités en batterie, véritable mémoire de masse pour toutes vos données et tous vos programmes.
C'est cela que vous attendiez !... alors, allez-y, maintenant !
MICRO DISC
ORIC
Dans le fond, vous avez eu raison d'attendre.
Maintenant vous pouvez faire le choix définitif. Voyez : mieux qu'un ordinateur personnel, ORIC vous offre tout un système de hautes performances.
Puissant pour vous emmener de l'initiation au BASIC jusqu'à la création de progiciels de gestion (sans oublier tous les jeux !).
Fiable, ergonomique et élégant pour représenter l'informatique personnelle parvenue à sa meilleure maturité.
Accessible pour tous les budgets ; ce système ORIC ATMOS, c'est la façon de dire : "Bon, voilà ce qu'il faut pour aller de l'avant, en avoir pour son argent, et être tranquille longtemps... donc, allons-y maintenant.
ORIC
IMPORTE ET DISTRIBUE PAR : ORIC-FRANCE
Z.I. «La Haie Griselle» B.P. 48 - Télex: 204 996
94470 BOISSY-ST-LEGER
Région Sud : 20, rue Vitalis 13005 MARSEILLE
R.C. Corbeil 318 041 530.

MILLE ET UNE RAISONS DE SOUSCRIRE UN ABONNEMENT A MICR'ORIC

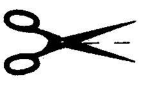
ORIC est le plus puissant des microordinateurs bon marché. MICR'ORIC est le meilleur magazine entièrement consacré à ORIC, sa technique, ses périphériques, sa programmation BASIC ou langage machine.

Numéro après numéro, MICR'ORIC vous conduira à une maîtrise approfondie de votre ORIC. Toujours bien informé, fourmillant d'idées originales grâce à la collaboration enthousiaste de nombre d'entre vous, il vous offre un recueil d'idées et de programmes très variés.

*L'abonnement est fixé à **100 F** pour 4 numéros ou **150 F** pour 6 numéros.*

Rythme de publication prévu : 6 numéros par an.

Pour toute correspondance indiquer votre numéro d'abonné, en particulier pour un changement d'adresse (gratuit).



BULLETIN D'ABONNEMENT

☐ *Je m'abonne aux 4 prochains numéros de MICR'ORIC* .... **100 F**

☐ *Je m'abonne aux 6 prochains numéros de MICR'ORIC* .... **150 F**

(préciser ici à partir du n° ________)

Pour compléter ma collection, je désire recevoir

les numéros suivants : ________ *à* ***40 F pièce*** *port compris, soit :* ________

(n° 1 et n° 2 épuisés)

Ci-joint un chèque total de : ________

à l'ordre de ***MICR'ORIC, Z.I. La Haie Griselle, B.P. 48, 94470 Boissy-Saint-Léger***

NOM : ________ *Prénom :* ________

Adresse : ________

Ville : ________ *Code postal :* |_|_|_|_|_|

Date : ________ *Signature :*

(des parents pour les mineurs)



LE MAGAZINE DES UTILISATEURS D'ORIC

SOMMAIRE

N°6

MICR'ORIC est une publication d'ORIC FRANCE, département de la société A.S.N. Diffusion

Directeur : Denis TAIEB

Rédacteur en chef : Lucien AUGUSTONI

Ont collaboré à ce numéro :
Pierre CHICOURRAT
Fabrice BROCHE
Gilbert FLAHAUT
Philippe BRAX
Marcel MORIN
P. GUILLERME
Max HAGENBURGER
Christophe ANDREANI
Marc BELLŒIL
Hervé LACOMBE

B.D. de la revue anglaise ORIC OWNER de Tansoft Ltd.

Adresse :
MICR'ORIC
Z.I. La Haie Griselle
B.P. 48
94470 Boissy-St-Léger

Dessins : Gilles TOCUT
Alain TALVAT

1re couverture, dessins, créations et conception :
STUDIO MELUN-IMPRESSIONS

Imprimerie :
MELUN-IMPRESSIONS,
18-19, rue E.-Briais, 77000 Melun
Tél. : (6) 452.04.31

Tirage : 10 000 exemplaires



LES COLONNES D'ORIC

Une fois de plus les fanatiques d'ORIC se déchaînent. Les trouvailles des uns et des autres s'additionnent, l'ORIC cessera-t-il d'avoir des secrets? Il semble que non, le numéro 7 vous apportera encore des surprises! A peine 10 % des possesseurs de matériel ORIC se procurent notre revue, c'est dommage pour les autres. Certains ont hésité à s'abonner. D'autres souhaitent se le procurer numéro après numéro, non sans avoir jeté un coup d'œil à l'intérieur auparavant. (On n'achète pas un chat dans un sac dit le proverbe!). Ils ne le trouvent pas facilement. La diffusion par le réseau habituel pose des problèmes. Si vous voulez encourager votre revue faites-là connaître autour de vous.

Quelle orientation intéressante pouvons nous annoncer? L'amélioration spectaculaire du D.O.S., un BASIC augmenté, un clavier de l'ATMOS dynamisé par la touche de fonction et tout ceci par de jeunes français virtuose du clavier. La diffusion de ces trouvailles est à l'étude. En outre, il est facile de comprendre qu'on s'achemine vers la production de programmes d'exploitation professionnelle du système

Ceux qui sont allés au SICOB BOUTIQUE ont peut être eu la chance de voir une démonstration d'un nouveau crayon optique.

Nous attendons les lecteurs-esclaves, le nouvel ORIC dont le nom "STRATOS" a été annoncé. Nous pensons qu'il sera compatible avec l'ATMOS et qu'il ne le supplantera pas, ce sera simplement un modèle différent possédant quelques particularités supplémentaires intéressantes.

ORIC FRANCE souhaite encourager les clubs informatiques existants. Envoyez-nous des documents (statuts, bulletins de liaison, catalogue d'activités...) et nous vous ferons connaître. D'autre part nous offrons à chacun des clubs ainsi recensés un service gratuit de la revue MICR'ORIC. Exemple : dans la région du Mans, connaissez-vous l'A.M.I.S., 1, rue de Moscou, 72190 Coulaines? Activités le mardi soir.

Dans le domaine des nouveautés, nous avons remarqué CHESS de IJK un jeu d'échec avec beaucoup de possibilités, CONTRE-ATTAQUE chez MP5 qui est un "DEFENCE FORCE" bon jeu d'arcade. On annonce la diffusion prochaine de "WIZARD of ORIC", jeu de rôles, gagnant de notre concours.

D'autres programmes fleurissent régulièrement, demandez des démonstrations chez les revendeurs.

Certains d'entre-vous n'ont pas su tirer parti du "SUPER D.O.S." de Denis Sebbag. Sachez que vous devez formatter votre disquette avec ce "SUPER D.O.S." et après avoir tapé !CONF1. Nous donnerons des informations plus complètes dans notre numéro 7, ceux qui sont pressés peuvent nous écrire, nous les renseigneront.

La suite de l'article "LES VARIABLES" de Pierre LEDAIN est reportée au numéro 7 faute de place.

Les amateurs d'ORIC sont de plus en plus nombreux, les collaborateurs de MICR'ORIC aussi, la passion est communicative, continuez ainsi, vous êtes formidables!

MICR'ORIC



VISITE EN TÊTE DE RAM

guidée par Pierre CHICOURRAT

Dans le numéro 4, Fabrice Broche vous a fait part de ses découvertes. Sur le même sujet voici les trouvailles de Pierre Chicourrat, ses commentaires, ses suggestions. Il s'agit ici de l'ORIC-1 (V 1.Ø), la transposition à l'ATMOS est possible, en particulier en se référant à notre n° 4. Les adresses sont en hexadécimal, le symbole ⁒ devant un nombre indique que, par exception, il s'agit d'un décimal.

(nn) contenu de l'adresse nn

(nn, nn+1) nombre codé sur 2 octets aux adresses nn et nn+1 : (nn, nn+1) = DEEK (nn)

nn —>pp toutes les adresses depuis nn jusqu'à pp

ØØ —> ØB : INUTILISÉES

ØC—> ØD : Variables de travail utilisées par l'interpréteur BASIC

(1Ø, 11) : Adresse du curseur sur l'écran HIRES on a; X et Y étant les coordonnées du curseur : (1Ø, 11) = AØØØ+4Ø*Y+INT(X/6)

(12, 13) : Adresse de l'endroit ou sera affiché le prochain message sur l'écran TEXT
par EX :
DOKE# 12,48Ø34:?".OK."
affiche '.OK.' à la place de 'CAPS' car ⁒48Ø34 est l'adresse de CAPS sur l'écran TEXT.

(18, 19) : Variable de travail utilisée par EDIT et LIST pour connaître l'adresse donnant le texte de l'ordre à afficher.
Si le code de l'ordre est ⁒128 (cas de END) (18, 19) au moment de l'affichage de cet ordre contiendra l'adresse en mémoire morte où est écrit ce qui correspond à ce code (soit ici 'END').

1A, 1B, 1C : Contiennent respect. les valeurs : 4C, ED, CB; soit en langage machine ce qui correspond au mnémonique :
JMP CBED
L'interpréteur fait un saut à cette adresse pour afficher le message d'invite : 'Ready' ces octets peuvent être modifiés pour détourner le micro-processeur juste avant l'affichage du message d'invite (par exemple pour remplacer ce message par 'JE SUIS PRÊT' rentrez le programme suivant et lancez le :

```
1Ø RESTORE:AD=# 4ØØ' ADRESSE IMPLANTATION
2Ø REPEAT:READI$:I=VAL("  "+I$):
POKEAD,I:AD=AD+1:UNTILI$="***"
3ØREM DONNÉES DU PROGRAMME L.M
4ØREM
5Ø DATA A9,Ø7,AØ,Ø4,4C,ED,CB
6Ø DATA 4A,65,2Ø,73,75,69,75,2Ø,7Ø,72,
65,74,ØD,ØA 'CODES ASCII DU MESSAGE
7Ø DATAØØ' OCTET NUL SIGNALANT LA FIN DU MESSAGE
8Ø DATA***' FIN DU PROGRAMME EN L.M
1ØØ DOKE# 1B,# 400' MISE EN SERVICE DE LA ROUTINE
```

1D, 1E : Variables de travail : utilisées par la fonction DOKE pour connaître l'adresse sur laquelle on travaille. Après utilisation de DOKE,(1D,1E) reste inchangé.
Ainsi :
DOKE# 65ØØ,12:?HEX$(DEEK(# 1D))
Retourne à l'affichage :
65ØØ
car le dernier DOKE avant lecture de (1D,1E) a été fait en 65ØØ
Ces 2 adresses peuvent donc servir à connaître l'adresse du dernier DOKE effectué.

1F, 20 : Variable de travail utilisée par la routine correspondant à PLOT et à LORES Ø (ou 1) (1F,2Ø) contient :
— dans le cas de PLOT l'adresse début d'affichage

— dans le cas de LORES l'adresse du bord supérieur gauche de l'écran.

21, 22, 23 : Contient en langage machine l'instruction de saut à la routine définie par 'DEFUSR'

24, 25 : Variables de travail : utilisées pour contenir des caractères de comparaison lors de certaines routines (peut être utilisé dans un de vos programmes en L.M)

27 : Contient le code ASCII du dernier caractère affiche (ou imprimé)

28, 29 : Mémoires de travail : dans le cas de manipulation de variables indique le type de variable manipulée :
Si (28)=FF et (29)=ØØ :
variable alpha-num.
Si (28)=ØØ et (29)=8Ø :
variable entière
Si (28)=(29)=ØØ :
variable réelle

2E : Si le bit7 de (2E) est à 1 tout affichage (ou impression) est supprimé; mais la machine continue, de plus le curseur sur l'écran ne bouge pas (comme si rien n'était affiché).
(2E) est restauré à Ø chaque fois que l'utilisateur reprend la main complètement (quand la machine affiche 'Ready')

30 : Nombre de caractères déjà imprimés sur la ligne courante, augmenté de %13 (en V1.Ø).
A chaque nouveau caractère imprimé cette mémoire augmente de 1 dès qu'elle atteint la valeur de (31) l'ORIC émet automatiquement un retour chariot.
Cette variable est aussi utilisée pour TAB; ce qui explique que l'argument de TAB doive être majoré de %13 pour fonctionner correctement (puisque quand (3Ø)=%13 il n'y a aucun caractère affiché).
On peut remarquer que :
POKE#3Ø,Ø:?TAB(12)"SALUT"
Fonctionne parfaitement : l'argument de TAB redevenant correct. Malheureusement à chaque passage à la ligne, (3Ø) est restauré à %13.

31 : Nombre maximum de caractères par ligne majoré de D
théoriquement on a donc au maximum %255-%13=%242 caractères pouvant être affichés avant que l'ORIC n'envoie son retour chariot.
Cependant si on charge (31) à %13 on dispose alors de %255 caractères.
Et d'une manière plus générale si on place dans 31 un nombre n inférieur à %13 on dispose de %255-n caractères par ligne. La limite des %242 caractères/l peut donc être dépassée.

33, 34 : Accumulateur pour les entiers codés sur 2 octets :
sert à stocker les numéros de lignes lors de l'entrée de ces lignes.
D'une manière générale toute routine de l'interpréteur fournissant comme résultat un entier sur 2 octets renvoie son résultat en (33, 34).

35 ——> 73 : Tampon clavier : lorsqu'une chaîne est entrée (par exemple lors de la saisie d'un programme), elle est d'abord stockée dans ce tampon.
La fin de la chaîne est signalée par un octet nul.
La routine se chargeant de la saisie des chaînes alphanumériques est en MEM à partir de l'adresse C5A2.
Ce tampon est aussi utilisé lors des opérations de sauvegarde pour stocker certaines données :
en (5F,60) adresse départ sauvegarde
en (61,62) adresse fin sauvegarde
en (67) vitesse :
si = Ø : 24ØØ bauds
si = 1 : 3ØØ bauds
en (64) type de programme :
si = Ø : basic
si <> Ø : langage machine
en (63) mode :
si = 1 : démarrage auto
si = Ø : pas de démarrage

53 ——> 70 : Nom du programme. La fin est signalée par un octet nul.

(9A, 9B) : Adresse bas du basic. La modification de cette adresse permet de protéger une zone avant le programme BASIC.
Ex. : DOKE# 9A, # 6ØØ:POKE# 5FF,Ø:NEW
Et le programme basic commencera à partir de 6ØØ. La zone de 5ØØ à 5FE est libre et protégée.
Attention ! Il est nécessaire que l'octet précédant le début du programme BASIC soit à Ø (ceci explique le 'POKE#5FF,Ø' de l'exemple).

(9C, 9D) : Adresse fin du programme basic.
La modification de (9A, 9B) et (9C, 9D) après un NEW permet de récupérer le programme.

(9E, 9F) : Adresse début de stockage des variables tableaux.
En fait elle indique la fin de la zone de stockage des variables numériques simples.
La modification de (9E, 9F) après un CLEAR (plus exactement sa restauration) permet de récupérer le contenu des variables par ex. :
DOKEØ,DEEK(# 9E):CLEAR:DOKE#9E, DEEK(Ø)
Entraîne l'effacement de toutes les variables sauf des variables réelles simples (qui sont

restaurées par le DOKE#9E,....

(AØ,A1) : Adresse fin de zone de stockage des variables simples et des tableaux.
Les tableaux à proprement parler sont donc stockés entre (9E, 9F) et (AØ, A1).
Pour les restaurer après un effacement total (CLEAR) il faut donc restaurer à la fois (9E, 9F) et (AØ, A1).

(A2, A3) : Adresse début de stockage du contenu des variables alpha-numériques.
Les variables alpha-numériques sont en effet stockées de façon particulière :
— dans la zone (9E, 9F) à (AØ, A1) on trouve les identificateurs de ces variables, c'est-à-dire :
— leur nom (2 caractères)
— leur longueur
— l'adresse où est stocké leur contenu
— dans la zone entre (A2, A3) et le HIMEM on trouve leur contenu.
A noter que le contenu des variables alphanumériques est donc stocké en fin de mémoire (ou du moins jusqu'au HIMEM).
Ceci explique que sur ORIC-1, lorsqu'un programme utilise la haute résolution, et si le HIMEM n'a pas été modifié, il arrive que le jeu de caractères se modifie : le HIMEM est en effet mal initialisé (bug de la ROM), et si le programme manipule des variables alphanumériques, leur contenu est stocké là où sont définis les caractères.
Un remède : mettre en début de programme :
HIMEM# 97FF
Et cet inconvénient disparaît.

(A4, A5) : Recopie de (A6, A7).

(A6, A7) : Adresse haut de la mémoire. C'est là qu'est stockée l'adresse indiquée après l'ordre HIMEM.

(A8, A9) : Numéro de la ligne en cours d'exécution.
De plus l'octet A9 indique à la machine si elle est en mode direct ((A9)=FF) ou en mode programme ((A9)<>FF).

Ceci explique que le numéro maximal accepté pour une ligne du programme ne soit pas 65535 (=FFFF) mais FEFF. En effet si une telle ligne existait, elle ne pourrait être correctement exécutée : la machine se croirait en mode direct et refuserait les fonctions telles que GET ou encore INPUT.
De plus si en mode direct on veut exceptionnellement utiliser des fonctions réservée au mode programme (GET ou INPUT) on peut modifier (A9) de manière à tromper la machine. Ainsi si l'on tape POKE# A9,1 , la machine se croira en mode programme même si elle est en mode direct.
Donc, en mode direct :
POKE# A9,1:GETRS
marche parfaitement.
Cependant il peut y avoir quelques problèmes, ces fonctions modifiant le tampon clavier...

(AA, AB) : En cas de BREAK contient le numéro de la ligne où s'est effectué le BREAK.

(AC, AD) : En cas de BREAK contient l'adresse où s'est arrêtée l'interprétation du programme.
Lorsque la machine rencontre à l'intérieur d'un programme un END elle réagit comme si c'était un BREAK. Après un END on peut donc faire un CONT.
De plus on peut tromper la machine et l'obliger à continuer (par CONT) alors qu'elle ne s'était pas arrêtée, tout simplement en chargeant dans (AC, AD) l'adresse à laquelle elle doit continuer, et en mettant dans (AA, AB) le numéro de la ligne à laquelle elle va continuer.
A ce propos pour trouver l'adresse à laquelle débute une ligne de numéro donnée, on peut utiliser (en mode direct) les commandes suivantes :
I=DEEK(# 9A):REPEAT:I=DEEK(I):U
NTILDEEK(I+2)=N
Ou on remplacera N par le numéro de la ligne considérée.

(BØ, B1) : Pointeur des DATA. On peut le modifier pour simuler un RESTORE à une ligne quelconque en faisant :
DOKE# BØ, AD
Ou AD est l'adresse début de la ligne où on veut faire le RESTORE (cette adresse peut être déterminée par la méthode indiquée ci-dessus).

(B4, B5) : Variable de travail pour l'interpréteur. Contient le nom de la variable sur laquelle on veut travailler (utilisée par exemple par la fonction READ).

(B8, B9) : Variable de travail pour l'interpréteur. Contient l'adresse du contenu de la variable sur laquelle on travaille (utilisée par exemple par la fonction READ).

BA ——> CF : Variables utilitaires pour l'interpréteur BASIC.

DØ ——> D4 : Accumulateur en virgule flottante : tout nombre calculé par l'interpréteur, s'il l'est en virgule flottante, est d'abord stocké ici : il s'agit en fait d'une mémoire tampon.

(D5) : Est utilisé pour indiquer le signe du nombre stocké dans l'accumulateur en virgule flottante.

E2 ——> F2 : Routine utilisée par l'interpréteur

BASIC lorsqu'il passe au caractère suivant à interpréter (routine GETCAR).

(E9, EA) : Adresse du prochain caractère à interpréter. Peut servir à détourner l'exécution d'un programme. Peut aussi permettre d'exécuter une partie de programme qui serait située ailleurs en mémoire.

Par ex. :

DOKE#E9,AD

Détourne l'exécution du programme à l'adresse AD.

FA ——> FE : Contient, en virgule flottante le nombre généré par la fonction RND.

La modification de ces 5 octets peut permettre de simuler la fonction RANDOMIZE.

En procédant par EX comme ceci :

DOKE#FB,DEEK(#276):DOKE#FD,DEEK(#277)

Cf signification de 276 et 277 ci-après.

200 ——> 2ø7 : Variables de travail utilisées par l'interpréteur pour la manipulation des paramètres des fonctions de son et de dessin.

208 : Code les touches frappées au clavier.

Tant qu'une touche est frappée, cette mémoire contient le code de la touche frappée, dès qu'elle est relâchée (2ø8) revient à 38 soit %56 en décimal.

Pour connaître le code d'une touche donnée vous pouvez utiliser le petit programme suivant :

```
1ØREPEAT:P=PEEK(# 2Ø8):K$=KEY$:
UNTILK$<>""
20?"CODE DE "K$" ="P
3Ø GOTO1Ø
```

Lancez ce programme et tapez la touche dont vous voulez connaître le code, vous pourrez ainsi constituer un tableau des codes du clavier.

Le remplacement des KEY$ d'un programme par la lecture de l'octet 2ø8 a un avantage certain : il apporte de la régularité dans les actions, en effet (2ø8) code réellement la touche frappée au moment de la lecture, alors que par KEY$ on n'a que la touche qui est prise en compte par la machine, et donc KEY$ dépend directement de la vitesse de répétition du clavier.

Pour les jeux on aura donc intérêt à remplacer les KEY$ par des PEEK(#2ø8) judicieux.

2ø9 : Code les touches CTRL et SHIFT non codées en 2ø8.

Le principe est le même que pour 2ø8 mais :

CTRL a pour valeur........................ A2
SHIFT gauche.............................. A4
SHIFT droite.............................. A7

Et sur ATMOS :

FUNCT..................................... A5

2ØA : Code le clavier.

2ØC : Indique si le clavier est en majuscule ((2ØC)=7F) ou en minuscule ((2ØC)=FF).

Si dans 2ØC vous mettez une valeur différente de 7F et FF vous changez la signification des touches (un A n'est plus affiché ni pris en compte comme un A).

Essayez par exemple :

POKE#2ØC,233

Et manipulez ensuite le clavier.

Bizarre non ??!!!!.....

Un RESET rétablit la situation.

212 : Quand on utilise une commande de trace de la couleur demandée (dernier paramètre pour toutes les fonctions de trace en HIRES) multipliée par %64.

213 : Contient le motif dessiné par DRAW ou CIRCLE, c'est-à-dire celui que vous pouvez sélectionner par PATTERN.

Ainsi :

POKE#213,N

Et :

PATTERN N

Ont le même rôle.

219, 21A : Coordonnées (X et Y) du curseur haute résolution.

Très utile si on ne sait plus où on en est.

21F : Indique l'endroit où sont définis les caractères :

Si (21F)=Ø :
même endroit qu'en mode TEXT
Si (21F)<>Ø :
même endroit qu'en mode HIRES.

228 : JMP à la routine de traitement des interruptions masquables.

A chaque interruption la machine fait un saut à l'adresse 228 où elle trouve un JMP à la routine de scrutation du clavier, etc...

Cette adresse n'est valable que sur les premières version de l'ORIC-1, mais elle est donnée sur les autres (et d'une manière générale sur toute machine architecturée autour d'un micro-pro 65Ø2) par :

?HEX$(DEEK(#FFFE))

22B : Adresse de JMP à la routine des interruptions non masquables NMI (correspondant au RESET).

Lorsqu'on appuie sur le bouton de RESET la machine arrête tout et fait un saut en 22B où elle trouve les mnémoniques correspondant au JMP vers la routine traitant le RESET.

Cette adresse n'est valable que sur ORIC-1, (Cf remarque ci-dessus).

Pour la trouver taper :

?HEX$(DEEK(#FFFA))

Et pour trouver l'adresse de la routine de RESET faire :
?HEX$(DEEK(DEEK(#FFFA)+1))

23Ø : Après avoir exécuté la routine de traitement des interruptions masquables (IRQ) la machine revient en 23Ø où elle trouve normalement le code 4Ø soit le mnémonique RTI, à moins que vous n'ayez détourné les interruptions pour une de vos applications.
Sur ATMOS (V 1.1) la même chose se trouve en 24A.

231 ——> 258 : Inutilisés : peut contenir une petite routine en L.M

(261) : Contient l'adresse à laquelle la machine doit faire un JMP lorsque vous faites un CTRL pour exécuter ce CTRL.
(En fait cette adresse est codée en (261, 262).

(264) : Code le scrolling vertical :
Si (264)=%26 : pas de scrolling
Si (264)=%27 : scrolling

(268) : Ordonnée (Y) du curseur TEXT

(269) : Abscisse (X) du curseur TEXT

(26A) : Code les CTRL (par exemple un CTRL D modifie l'état (par une inversion) du bit b6.
Chaque CTRL (sauf CTRL T) modifie ainsi l'état d'un bit de 26A (voit MICR'ORIC n° 4).

26B, 26C : Contient respectivement le caractère correspondant à la couleur du papier et de l'écran.
Contient en fait le code ASCII des caractères qui sont mis dans les deux premières colonnes de l'écran TEXT.
Essayez par EX :
POKE#26B,64:POKE#26C,65:CLS

(26D, 26E) : Adresse début de l'écran TEXT ou HIRES.
Une modification de cette valeur peut permettre de définir une fenêtre d'écran.

(26F) : Nombre de ligne de l'écran TEXT
Essayez par EX :
POKE#26F,5:CLS
Et voyez le résultat...
Pour rétablir la situation soit RESET soit :
POKE#26F,27

(272, 273) : Compteurs décrémentés à chaque
(274. 275) interruption du micro-processeur.
(276, 277) : Peuvent servir d'horloge au 100e de seconde pour mesurer le temps séparant 2 évènements.
Peuvent servir de chrono (vraiment très précis).

2CØ : Indique le mode dans lequel on travaille en fonction de la valeur de ses 3 premiers bit.
Si bit Ø et bit 1 sont nuls :
mode GRAB
Si bit Ø et bit 1 forcés à 1 :
modes HIRES
Si bit Ø = Ø et bit 1 = 1 :
mode TEXT

2DF : Contient le code ASCII de la dernière touche frappée augmente de %128=80
Si (2DF)=Ø : pas de touches frappées.

2EØ : Indique une erreur dans la valeur des paramètres transmis après une instruction de tracé en HIRES (si (2EØ)<>Ø : erreur).

2E1 ——> 2E6 : Tampon de transmission des paramètres pour les instructions de son, de dessin, et commandant l'affichage (encre ou papier).
Par ex. si vous faites :
CURSET12Ø,1ØØ,1
Alors :
(2E1,2E2)=12Ø
2E3,2E4,)=1ØØ
(2E5,2E6)=ØØ1

2F1 : Si le bit de (2F1) est à 1 tout caractère devant être affiché est renvoyé à l'imprimante.
Par ex. :
POKE#2F1,128:?"OK:ÇA MARCHE"
Équivaut à :
LPRINT"OK:ÇA MARCHE"
(2F1) est restauré à Ø quand l'utilisateur reprend la main (affichage de 'Ready').
Très utile pour adapter immédiatement un programme à l'imprimante.

2F2 : Si le bit 7 de (2F1) est à 1 indique que l'on fait un EDIT. cette mémoire est utilisée par les routines en ROM (en C799 et C7EB) de l'interpréteur.

2F4 : Si le bit 7 est à 1 indique que l'on est en TRON sinon TROFF.
Pour remettre un programme en TRON après un arrêt faire :
TRON:CONT
Donc TRON est utilisable aussi en **mode direct.**

2F7 : Permet de modifier la définition du clavier : ce qui est affiché n'étant plus ce qui est pris en compte.
Essayez par exemple :
POKE#2F7,32
Et réfléchissez sur le résultat : qui a dit que le BASIC ne comprenait pas les minuscules Adresses utilisables sur ORIC-1 exclusivement (peut être un équivalent sur ATMOS ???).

(suite et fin page 53)

DOMINEZ VOTRE CLAVIER

par Fabrice BROCHE



La gestion du clavier de l'ORIC présente quelques inconvénients parmi lesquels on peut citer :

* 20 % du temps est passé à gérer le clavier, la plupart du temps pour rien.
* Impossible de savoir si deux touches sont pressées en même temps et a fortiori lesquelles...
* La fonction KEY$, est peu souple d'emploi du fait qu'elle est de type alphanumérique.

Nous allons voir comment nous pouvons très souvent nous affranchir de ces inconvénients, en expliquant comment gérer le clavier. Rien de bien compliqué, rassurez-vous ; même si vous ne connaissez rien en assembleur cet article vous sera utile.

Pour commencer examinons le principe de base

Le clavier de l'ORIC est en forme de **matrice**, c'est-à-dire une combinaison lignes X colonnes. La pression sur une touche ne fait que mettre en contact une ligne et une colonne comme le montre le schéma que voici :

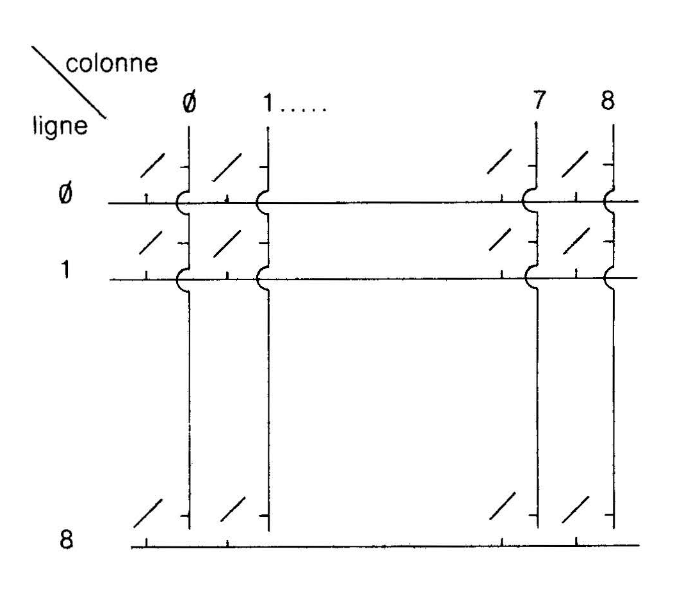


SCHÉMA 1

Une touche correspond donc à une intersection : aussi pour savoir si l'on appuie sur une touche donnée, il faut tester une ligne et une colonne précises.

Pour cela nous mettons la colonne sélectionnée au niveau logique 1 par exemple, les lignes étant à l'origine à Ø. En sondant la ligne concernée on s'aperçoit que la touche a été actionnée si elle est passée à 1.

Ainsi, on saura parfaitement quelles sont les touches enfoncées si pour chacune des 8 colonnes on balaye les 8 lignes. Mais, nous n'en sommes pas encore là. Vérifions ce qui vient d'être affirmé.

Voici le programme pour ATMOS, avec indication des changements à opérer pour ORIC-1.

```
0400 08     PHP
0401 78     SEI        Supprimer la gestion normale.
0402 A900   LDA %#00
0404 8500   STA #00    00 pointeur des colonnes
0406 A27F   LDX %#7F
0408 8A     TXA        première colonne activée No 7.
0409 48     PHA        Sauvegarder masque colonne dans la pile.
040A A90E   LDA %#0E
040C 8D0F03 STA #030F
040F A9EE   LDA %#EE
0411 8D0C03 STA #030C
0414 A9CC   LDA %#CC
0416 8D0C03 STA #030C
0419 8E0F03 STX #030F
041C A9EC   LDA %#EC
041E 8D0C03 STA #030C
0421 A9CC   LDA %#CC
0423 8D0C03 STA #030C  Activation colonne.
0426 A000   LDY %#00   Y contient le 'DESSIN' d'une colonne. 0 au début.
0428 A207   LDX %#07   Boucler sur toutes les lignes.
042A 8A     TXA
042B 0910   ORA %#10   Ne pas activer l'imprimante.
042D 8D0003 STA #0300  Activation.
0430 A908   LDA %#08
0432 2C0003 BIT #0300  et test de la ligne.
0435 F004   BEQ #043B
0437 98     TYA        Si connectée
0438 15F8   ORA #F8,X
043A A8     TAY        Actualiser 'DESSIN' d'une colonne.
043B CA     DEX
043C 10EC   BPL #042A  Boucler sur les lignes.
043E A600   LDX #00    X=No colonne.
0440 E600   INC #00    Incrémenter colonne.
0442 9404   STY #04,X  Et sauver 'DESSIN' colonne actuelle.
0444 68     PLA        Restaurer masque colonne.
0445 38     SEC        Permutation circulaire d'un cran à droite.
0446 6A     ROR A
0447 AA     TAX        et passage à la colonne suivante.
0448 B0BE   BCS #0408  Si un '0 sort - c'est la fin.
044A 28     PLP
044B A91E   LDA %#1E            AFFICHAGE
044D 20D9CC JSR #CCD9  #CC12 sur ORIC 1
0450 A207   LDX %#07   Curseur en haut à gauche.
0452 A007   LDY %#07
0454 1604   ASL #04,X
0456 9003   BCC #045B
0458 A930   LDA %#30
045A 2CA931 BIT #31A9
045D 20D9CC JSR #CCD9  #CC12 sur ORIC 1
0460 88     DEY
```

```
0461 10F1   BPL #0454
0463 20F0CB JSR #CBF0    #CB9F sur ORIC 1
0466 CA     DEX
0467 10E9   BPL #0452
0469 60     RTS
```

Pour l'utiliser POKER en Ø un numéro de ligne et en 1 un numéro de colonne. Lancer le programme par CALL # 4ØØ. Après exécution un PRINT PEEK (2) vous donnera 8 ou Ø selon que la touche était enfoncée ou non.

Les plus curieux d'entre vous ont sans doute déjà dans leurs documents une table de la matrice du clavier. Il auront pu s'apercevoir que les colonnes correspondent assez bien à la position des touches en colonnes mais que, par contre, pour les lignes il n'y a pas correspondance. Qui a compris pourquoi ?

Mais vous êtes curieux de savoir ce qui se passe effectivement
Comment active-t-on une colonne ?
Comment teste-t-on une ligne ?

Encore un peu de théorie

Considérons toujours le clavier comme un ensemble d'intersections entre 8 colonnes et 8 lignes. La première idée qui vient à l'esprit est d'utiliser 1 bit pour chaque colonne et 1 bit pour chaque ligne. Avec un microprocesseur 8 bits il y a un problème, il faut aussi penser à la gestion du magnétophone...

La solution réside en un **démultiplexage.** Voyons en quoi cela consiste.

Si l'on voulait connecter chaque ligne (rangée) de touches du clavier à une ligne d'entrée/sortie, il y aurait un gaspillage énorme : alors qu'un octet peu prendre 256 valeurs distinctes, nous n'en utiliserions que 8 (1, 2, 4, 8, 16, 32, 64, 128) alors qu'un nombre de Ø à 7 ferait aussi bien l'affaire, avec le gros avantage de pouvoir se coder sur 3 bits seulement. Plus de gaspillage, on économise 5 lignes d'entrée/sortie.

Il n'empêche qu'à la sortie du clavier les lignes sont bien au nombre de 8. C'est ici qu'intervient le **démultiplexage.** Il va transformer le numéro de ligne (Ø à 7) en un masque sur 8 bits qui, lui, prendra les valeurs 1, 2, 4... 128.

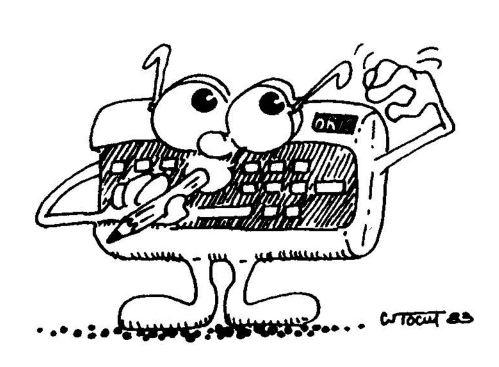
Notez que le démultiplexage des colonnes n'a pas été nécessaire car les 5 lignes d'entrée/sortie libérées suffisent amplement à l'ORIC. Voilà un premier aspect de la question.

Attachons nous de plus près à la structure des entrées/sorties.

L'ORIC comporte un processeur spécialisé, le très connu VIA 6522 A. Un tel processeur se prête à de multiples utilisations que nous pourrons examiner dans un article ultérieurement. Limitons nous ici à ce qui permet le dialogue avec le clavier.

Ce circuit comporte 2 ports, A et B, chacun sur un octet. On peut, selon la configuration, lire ou écrire sur les lignes de ces ports. Si vous supposez le port A en sortie, et que vous y écrivez ØØØØØ111, les broches Ø, 1 et 2 du port A vont passer à 5 V, les autres vont rester à Ø V, c'est aussi simple que cela.

Mieux qu'un long discours, un schéma de l'environnement vous éclairera : les flèches indiquent la direction normale des données.



Nous voyons donc qu'il nous sera facile de tester les lignes. Il est un peu plus difficile d'activer les colonnes puisqu'il faudra passer par le 8912.

Décrivons ces opérations : le 8912 à 14 registres adressables. Deux seulement nous intéressent, le n° 7 qui contient la direction du port (entre autres), nous n'y toucherons pas ; il est normalement configuré en sortie. Plus intéressant est le registre n° 14 qui permet d'envoyer des données sur le port A du 8912, ces données ne faisant alors que **passer** par le 8912.

Nous allons voir une autre forme de multiplexage : toujours pour économiser des lignes d'entrée/sortie, et donc faciliter l'interfaçage, le bus d'adresses et le bus de données ont été ici réunis, c'est-à-dire qu'on envoie d'abord sur le bus le n° de registre qu'on veut lire ou écrire, qu'on fait savoir que c'est une adresse grâce aux lignes de contrôle CA2 et CB2, et qu'on envoie ensuite la donnée, avant de signaler là encore que c'est une donnée qui est présente sur le bus. Ainsi voici une procédure standard d'envoi de donnée.

```
0470 A601    LDX #01
0472 A9FF    LDA %#FF
0474 18      CLC
0475 2A      ROL A
0476 CA      DEX
0477 10FC    BPL #0475
0479 AA      TAX
047A A90E    LDA %#0E
047C 2090F5  JSR #F590     #F535 sur ORIC 1
047F A500    LDA #00
0481 8D0003  STA #0300     Activation de la colonne contenue en 1.
0484 A908    LDA %#08      Demande de tester la ligne contenue en 0.
0486 2D0003  AND #0300
0489 8502    STA #02       Si la ligne et la colonne sont connectées - Bit 3=1
048B 60      RTS
```

Bien entendu, une telle routine existe dans le BASIC, son utilisation est la suivante :

```
0490 08       PHP
0491 78       SEI           Pour que la gestion normale ne gene pas.
0492 20CBD8   JSR #D8CB     #D810 sur ORIC 1
0495 8A       TXA           X contient l'opérande de USR
0496 2938     AND %#38      Les bits 3-4-5 contiennent la ligne
0498 4A       LSR A         on les isole.
0499 4A       LSR A
049A 4A       LSR A         et on les déplace dans les bits 0-1-2
049B 48       PHA           puis on les sauve sur la pile.
049C 8A       TXA
049D 2907     AND %#07      Récupérer la colonne.
049F AA       TAX
04A0 18       CLC
04A1 A9FF     LDA %#FF      Démultiplexage de la colonne.
04A3 2A       ROL A
04A4 CA       DEX
04A5 10FC     BPL #04A3
04A7 AA       TAX           Dans X.
04A8 A90E     LDA %#0E
04AA 2090F5   JSR #F590     #F535 sur ORIC 1
04AD 68       PLA           Activation colonne.
04AE 0910     ORA %#10      Ne pas activer le STROBE imprimante
04B0 8D0003   STA #0300     mais... activer le démultiplexeur.
04B3 A908     LDA %#08      Masque de TEST + temps de réponse.
04B5 28       PLP
04B6 2C0003   BIT #0300
04B9 F003     BEQ #04BE     Si connectée... faire
04BB 4C0FDF   JMP #DF0F     un TRUE (-1)                #DF00 sur ORIC 1
04BE 4C0BDF   JMP #DF0B     sinon un FALSE (0)          #DEFC sur ORIC 1
```

De plus, pour activer une colonne, il suffit puisqu'elles ne sont pas multiplexées, de mettre à Ø le bit correspondant, tous les autres étant à 1. Exemple : activation de la colonne 6 : 1Ø111111.

Pour les lignes, la procédure est beaucoup plus simple : il suffit d'envoyer le n° de ligne en l'écrivant simplement sur le port B et de tester la réponse du multiplexeur qui met à 1 la ligne de réponse, soit PB3.

Attention, toutefois : d'une part écrire un nombre de Ø à 7 revient aussi, c'est évident, à mettre les lignes 4 à 7 à zéro. Il peut être utile de laisser la broche PB4, qui commande l'imprimante, à 1, pour ne pas activer inutilement l'imprimante.

D'autre part, la réponse du multiplexeur n'est pas immédiate : entre le moment où l'on envoie la ligne et le moment où on lit la réponse, il faut au moins une instruction.

Certains concepteurs de jeux l'ont négligé et leurs jeux ne fonctionnent pas avec tous les ORICs.

Voici une procédure standard d'entrée/sortie de ligne, valable pour un ORIC-1.

```
                 E/S avec le 8912.

LDA %#0E       Mettre registre n° 14 sur le bus de données.
STA #30F
LDA %#EE       CA2 et CB2 à 1 état bas
STA %#30C       produit la validation.
LDA donnée     Mettre la donnée sur le bus de données.
STA #30F
LDA %#CC       CA2 bas CB2 haut donc c'est une donnée.
STA #30C
LDA %#CC       Validation.
STA #30C
```

E/S avec le démultiplexeur.

```
LDA n° ligne
STA #300       Activation multiplexeur.
LDA %#08       Masque de test et... temps de réponse.
AND #300       Test Z=1 si la touche est pressée sinon Z=0.
```

E/S avec le 8912

```
LDA n° de registre
LDX donnée
JSR #F535   (#F590 pour l'Atmos)
```

M'avez-vous suivi? Si oui, bravo! Si non, lisez ce qui suit, cela va vous intéresser.

Pour bien voir clair dans tout ceci, élaborons un programme facile d'emploi par BASIC et qui sera très, très utile aux créateurs de jeux. Il permet de tester une touche précise, indépendamment de toutes les autres, et ce, très rapidement. Vous pouvez même supprimer les interruptions qui, normalement, gèrent le clavier.

L'idée est la suivante : grâce à la fonction USR, on propose une ligne et une colonne.

En retour, l'accumulateur contiendra la valeur TRUE (—1) si la touche correspondante est enfoncée, la valeur FALSE (Ø) dans les cas contraire.

La syntaxe est la suivante :

USR (colonne + 8 * ligne).

Il aurait été encore plus simple de spécifier le code ASCII, mais cela aurait nécessité la conversion via un tableau, ce qui éloigne du sujet. Et puisque le but de cet article est de vous amener à savoir faire ce genre de programme, nous en resterons là :

PROGRAMME 'Clavier 8 x 8'

```
PHP
SEI                Supprimer la gestion clavier de l'ORIC.
LDA  %#00
STA  #00           00 pointeur colonne.
LDX  %01111111 Masque pour activer colonne 7.
TXA                     COLONNE
PHA                Sauvegarde du masque dans la pile.
LDA  %#0E          Registre 14 du 8912 = port clavier colonnes.
STA  #30F
LDA  %#EE
STA  #30C          Valider valeur #30F comme No de registre.
LDA  %#CC
STA  #30C          Repos.
STX  #30F          Envoyer masque colonne.
LDA  %#EC
STA  %#30C          Valider valeur #30F comme donnée.
LDA  %#CC
STA  #30C           Repos
LDY %#00          Y contient le 'dessin' d'une colonne.
LDX  %#07          Boucler sur 8 lignes.
TXA                      LIGNE
ORA  %#10          Strobe imprimante à 1.
STA  #300          Valider la ligne vers le multiplexeur.
LDA  %#08          Le multiplexeur renvoie un 1.
BIT  #300          sur le bit 3 si la ligne et la colonne sont connectées.
BEQ  Suite         Si touche non enfoncée...saute.
```

```
TYA
ORA   #F8,X      Mettre un 1 en Xème position sur le
TAY                dessin de la colonne.
DEX
BPL   ligne      Test si toutes les lignes sont testées.
LDX   #00         X = n° de colonne.
INC   #00        Incrémenter colonne.
STY   #04,X       Sauver dessin de la Xème colonne.
PLA              Restaurer masque colonne.
SEL
ROR              Déplacer la colonne d'un cran vers la droite.
BCS   colonne    Si un 0 sort - Fin de recherche.
LDX   %#07       Pour chaque colonne.
LDY   %#07       Afficher chaque ligne.
ASL   #04,X       La valeur de la colonne sort dans la retenue.
BCC   écris 0    Astuce classique. Si le programme branche
LDA   '1'        ICI le programme sera  LDA '1'
BIT   #2C        BIT #00A9 ici instruction 'creuse'.
LDA   '0'
JSR   #CC12       Affichage de l'accumulateur.(#CCD9 sur ATMOS)
DEY
BPL   ligne      Test fin de ligne
JSR   #CB9F      Si changement de colonne...aller à la ligne.
DEX              (#CBF0 sur ATMOS)
BPL   colonne    Test fin de colonne.
PLP
LDA   %#1E
JMP   #CC12       Finir en renvoyant le curseur en 0.0
(#CCD9 sur ATMOS)

Temps d'exécution (sans affichage) 1814+7x microsecondes.
(x est le nombre de touches pressées)
```

Voici une table de "dématriçage"

LIGNE

COLONNE	7	6	5	4	3	2	1	Ø
Ø	8	Y	U	espace	K	M	J	7
1	L	H	I	,	9	6	T	N
2	Ø	G	O	.	;	B	R	5
3	/	E	P	↑	—	4	F	V
4	SHIFT droite		FUNCT	SHIFT gauche		CTRL		
5	RETURN	A	DEL	←		Z	ESC	1
6		S	]	↓		2	Q	X
7	=	W	[	→	'	C	D	3

La touche <RETURN> correspond à la 5[e] colonne et à la 7[e] ligne. Voici comment tester cette touche dans un programme BASIC :

```
1Ø DEF USR = ADRESSE
   (la routine peut être logée n'importe où)
2Ø PRINT USR (5 + 8 * 7)
3Ø GOTO 2Ø
```

L'examen de la table permet de bien comprendre pourquoi l'ORIC isole facilement les touches SHIFT, CTRL, FUNCT : il teste la colonne 4.

Vous vous demandez sans doute pourquoi les lignes et les colonnes sont inversées. Le dernier programme que nous vous proposons, qui est une synthèse de tout ce qui vient d'être exposé va visualiser à tout instant le clavier sous la forme ci-dessus. Un 1 s'affiche si la touche est pressée, sinon c'est un Ø. Quel plaisir de constater que, si l'on appuie sur **plusieurs** touches en même temps, l'ORIC les reconnaît enfin !

Programme compatible ORIC-1 - ATMOS

```
100 REM ============================
110 REM =                          =
120 REM =      GESTION CLAVIER     =
125 REM =      ORIC  V1.0 V1.1     =
130 REM =                          =
140 REM =   AUTEUR:Fabrice BROCHE  =
150 REM =        LE 12/03/84       =
160 REM =                          =
170 REM =     Ripelle Software     =
180 REM =                          =
190 REM ============================
200 REM
210 REM
300 REM
310 REM ============================
320 REM ====== ENTREE ROUTINE 1 =====
330 REM ============================
1000 FOR I=#400 TO #469
1010 READ A$:DT=VAL("#"+A$)
1020 POKEI,DT
1030 NEXT
1040 A=1
1045 REM ====== PLACER MASQUES ======
1050 FOR I=#F8 TO #FF
1060 POKE I,A:A=2*A
1070 NEXT I
1080 IF PEEK(#FFFE)=40 THEN 1160
1090 DOKE#44E,#CCD9
1100 DOKE#464,#CBF0
1110 DOKE#45E,#CCD9
1130 REM ==========================
1140 REM ===== ENTREE ROUTINE 2 =====
1150 REM ==========================
1160 FOR I=#470 TO #48B
1170 READ A$:DT=VAL("#"+A$)
1180 POKE I,DT
1190 NEXT
1200 IF PEEK(#FFFE)=40 THEN 1250
1210 DOKE#47D,#F590
1220 REM ==========================
1230 REM ===== ENTREE ROUTINE 3 =====
1240 REM ==========================
1250 FOR I=#490 TO #4C0
1260 READ A$:DT=VAL("#"+A$)
1270 POKE I,DT
1280 NEXT
1290 IF PEEK(#FFFE)=40 THEN 2000
1300 DOKE#493,#D8CB
1310 DOKE#4AB,#F590
1320 DOKE#4BC,#DF0F
1330 DOKE#4BF,#DF0B
1340 REM ==========================
1350 REM =  Représentation du cla-  =
1360 REM =  vier par une matrice    =
1370 REM =  8x8 ATTENTION :les li-  =
1380 REM =  gnes et les colonnes    =
1390 REM =  sont inversées...       =
1400 REM ==========================
2000 CLS:POKE#26A,10
2010 CALL#400
2020 GOTO2010
3000 REM ==========================
3010 REM =POKER en 0 un numéro de li=
3020 REM =gne,et en 1 un numéro de  =
3030 REM =colonne:après un CALL #470=
3040 REM =PEEK(2) retournera 8 ou 0 =
3050 REM =selon que la touche est ou=
3060 REM =non pressée .             =
3070 REM ==========================
3080 REM
3085 REM ======= TEST ESPACE =======
3090 POKE 0,4:POKE 1,0
3100 CALL #470
3110 PRINT PEEK(2)
3120 GOTO 3100
4000 REM ==========================
4010 REM = USR(colonne +8*ligne)    =
4020 REM = retourne la variable boo =
4030 REM = léenne TRUE/FALSE selon  =
4040 REM = que la touche est ou non =
4050 REM = pressée .                =
4060 REM ==========================
4070 DEF USR=#490
4080 REM ======= TEST ESPACE =======
4090 PRINT USR(0+8*4)
4100 GOTO 4090
8030 REM ==========================
8040 REM = ROUTINE  CLAVIER MATRICE =
8050 REM ==========================
9000 DATA 08,78,A9,00,85,00,A2,7F
9001 DATA 8A,48,A9,0E,8D,0F,03,A9
9002 DATA EE,8D,0C,03,A9,CC,8D,0C
9003 DATA 03,8E,0F,03,A9,EC,8D,0C
9004 DATA 03,A9,CC,8D,0C,03,A0,00
9005 DATA A2,07,8A,09,10,8D,00,03
9006 DATA A9,08,2C,00,03,F0,04,98
9007 DATA 15,F8,A8,CA,10,EC,A6,00
9008 DATA E6,00,94,04,68,38,6A,AA
9009 DATA B0,BE,28,A9,1E,20,12,CC
9010 DATA A2,07,A0,07,16,04,90,03
9011 DATA A9,30,2C,A9,31,20,12,CC
9012 DATA 88,10,F1,20,9F,CB,CA,10
9013 DATA E9,60
9014 REM ==========================
9015 REM ====== ROUTINE "TEST" =====
9016 REM ==========================
9017 DATA A6,01,A9,FF,18,2A,CA,10
9018 DATA FC,AA,A9,0E,20,35,F5,A5
9019 DATA 00,8D,00,03,A9,08,2D,00
9020 DATA 03,85,02,60
9021 REM ==========================
9022 REM ======= ROUTINE "USR" =====
9023 REM ==========================
9024 DATA 08,78,20,10,D8,8A,29,38
9025 DATA 4A,4A,4A,48,8A,29,07,AA
9026 DATA 18,A9,FF,2A,CA,10,FC,AA
9027 DATA A9,0E,20,35,F5,68,09,10
9028 DATA 8D,00,03,A9,08,28,2C,00
9029 DATA 03,F0,03,4C,00,DF,4C,FC
9030 DATA DE
9031 REM
9032 REM ==========================
9033 REM ==========================
```

Ceux qui proposeront désormais des jeux où il faut s'arrêter pour tirer et arrêter le tir pour se déplacer n'auront plus d'excuse.

DOMINEZ VOTRE CLAVIER APPLICATION

par Fabrice BROCHE

Puisque nous avons vu comment, en BASIC, détecter la pression de plusieurs touches à la fois, mettons en pratique un programme de jeu, sans prétention, où pour marquer des points, il faut appuyer sur au moins deux touches à la fois, si ce n'est trois.

Règles du jeu :

Vous êtes le **pavé blanc** et vous devez détruire les **pavés striés** qui apparaissent aléatoirement sur l'écran. Pour ce faire. vous disposez de 5 touches.

A déplacement vers la gauche.

S déplacement vers la droite.

O déplacement vers le haut.

K déplacement vers le bas.

barre d'espace : Attaque.

Pour effacer les **pavés striés** vous devez être en coïncidence et appuyer sur la barre d'espace, la forme de votre pavé est alors modifiée et vous marquez 100 points. Mais attention il ne faut pas que vous soyez immobile lors de l'attaque sinon vous perdez des points et votre score peut devenir négatif.

Bien entendu, les déplacements en diagonale sont possibles. A tout moment un appui sur la touche ESC rendra la main.

Ne nous attardons pas sur les subtilités de la marque ou sur les règles du jeu, vous les découvrirez facilement. Voyons, en revanche, de plus près le programme.

Pour obtenir un exécution assez rapide, quelques principes ont été utilisés. Principes qui se résument en ceci : "Mâcher le travail à l'ORIC". Et ce, sur les points suivants :

① Lorsque, pour un GOTO ou un GOSUB, l'ORIC recherche un n° de ligne, il commence son exploration au début du programme. On a donc mis en fin de programme les routines qui ne servent qu'une fois.

② Les nombres sont codés d'une telle façon sur ORIC que leur conversion demande un temps non négligeable. Si des valeurs, même constantes, sont souvent rencontrées, il ne faut pas hésiter à les affecter à des variables. C'est pourquoi, par exemple, le code de test de chaque touche a été calculé une fois pour toutes et affecté à des variables, ce qui, de plus, augmente la lisibilité du programme. (ligne 626Ø)

③ Lorsque 2 propositions A et B doivent être vérifiées en même temps pour donner C, vous avez deux méthodes :

a) IF A AND B THEN C

b) IF A THEN IF B THEN C

Il s'avère que la méthode b) est plus efficace : la condition B n'étant évaluée que si A est vraie, alors qu'avec la méthode a) les conditions A et B sont toutes deux testées à chaque fois.

Ce principe a été utilisé aux lignes 2Ø7Ø-21ØØ, en veillant à mettre la condition la moins probable en A. A la ligne 2Ø7Ø, par exemple, il est largement plus fréquent que X>L1 soit vrai plutôt que USR(A). (Appui sur A)

④ PLOT est plus rapide que POKE ou PRINT. Il a donc été préféré.

⑤ Puisque la gestion des touches se fait par USR ( ), il est inutile de perdre du temps avec la lecture du clavier effectué 100 fois par seconde. Un gain de 20 % en vitesse est obtenu grâce à POKE#3ØE, 127 (Inhibition des IRQ)
POKE#3ØE, 192 (Réactivation normale du clavier)

Espérons que les possibilités ici décrites vous permettes d'améliorer vos programmes passés et futurs et d'éblouir vos amis par votre virtuosité.

→

```
1000 REM============================
1010 REM=                          =
1020 REM=     ORIC  V1.0 V1.1      =
1030 REM=                          =
1040 REM=        DEMO CLAVIER      =
1050 REM=                          =
1060 REM=  AUTEUR:Fabrice BROCHE   =
1070 REM=       LE 01/07/84        =
1080 REM=                          =
1090 REM=     Ripelle Software     =
1100 REM=                          =
1110 REM============================
1120 REM
1130 REM
1140 REM
1150 GOSUB 6000
1160 REM
1170 REM
1180 REM============================
1190 REM=========== JEU ===========
1200 REM============================
1210 REM=========AFFICHAGE==========
2000 PLOT X,Y,B$
2010 X=XX:Y=YY
2020 PLOT X,Y,A$
2030 A$=D$
2040 REM========TEST ESPACE=========
2050 IF USR(SP) THEN A$=C$:SN=SN-1:PLOT
26,1,STR$(SN)+" ":IF SN=0 THEN 4000
2060 REM=========DEPLACEMENT========
2070 IF USR(A) THEN IF X>L1 THEN XX=X-DX
2080 IF USR(S) THEN IF X<L2 THEN XX=X+DX
2090 IF USR(O) THEN IF Y>L3 THEN YY=Y-DY
2100 IF USR(K) THEN IF Y<L4 THEN YY=Y+DY
2110 REM=====SORTIE PREMATUREE======
2120 IF USR(ESC) THEN 5000
2130 SC=SCRN(XX,YY)
2140 REM====TEST POUR SON VAISSEAU===
2150 IF SC=#7C THEN VA=VA-100:PLOT 9,1,S
TR$(VA)+"   "
2160 IF SC<>#26 THEN 2190
2170 REM=======ENNEMI TUE==========
2180 IF A$=C$ THEN VA=VA+100:PLOT 9,1,ST
R$(VA):NE=NE-1:IF NE=0 THEN 3000
2190 IF RND(1)>NN THEN 2000
2200 REM========AJOUT ENNEMI========
2210 PLOT INT(RND(1)*18)*2+2,INT(RND(1)*
12)*2+2,E$:NE=NE+1
2220 GOTO 2000
2230 REM============================
2240 REM====== FIN DE PARTIE =======
2250 REM============================
3000 AJ=10000:A$="VOUS AVEZ GAGNE !":GOT
O 3020
3010 AJ=0:A$="  C'EST FINI !!"
3020 CLS:PRINT:PRINT
3030 POKE#30E,192:POKE#26A,11
3040 PRINT A$:PRINT
3050 PRINT "VOTRE SCORE:":PRINT
3060 PRINT "    "AJ"   SUPER BONUS"
3070 PRINT " + "VA"    SCORE
3080 PRINT " + "10*SN"    SPACES RESTANTS
"
3090 PRINT "    -------"
3100 PRINT " = "AJ+VA+10*SN
3110 PRINT
3120 PRINT "UNE AUTRE PARTIE ?";
3130 WAIT 200
3140 REM=====VIDER TAMPON TOUCHE=====
3150 POKE #2DF,0
3160 GETA$:IFA$="O"THEN RUN
3170 GOTO 5000
3180 REM============================
3190 REM===== PLUS DE BOUCLIER ======
3200 REM============================
4000 NJ=NJ-1:IF NJ>0 THEN SN=80:PLOT 35,
1,STR$(NJ):PING:GOTO 2000
4010 GOTO 3010
4020 REM============================
4030 REM=========== FIN ===========
4040 REM============================
5000 POKE #30E,192:PAPER 0:INK 7:CLS
5010 POKE #26A,11
5020 END
5030 REM============================
5040 REM====== INITIALISATION.. =====
5050 REM============================
5060 REM
5070 REM===SUPRIMER IRQ:ACCELERER====
6000 POKE #30E,127
6010 REM==========ENTREE USR========
6020 FOR I=#400 TO #430
6030 READ A$:DT=VAL("#"+A$)
6040 POKE I,DT
6050 NEXT
6060 REM======MODIFIER SI ATMOS======
6070 IF PEEK(#FFFE)=40 THEN 6120
6080 DOKE #403,#D8CB
6090 DOKE #41B,#F590
6100 DOKE #42C,#DF0F
6110 DOKE #42F,#DF0B
6120 DEF USR=#400
6130 REM=======ENTREE DESSINS=======
6140 READ A$:X=#B400+8*ASC(A$):IF A$="FI
N" THEN 6180
6150 FOR X=X TO X+7:READ A:POKE X,A:NEXT
6160 GOTO 6140
6170 REM=========INIT CHAINES========
6180 B$="  ":C$="ùù":D$="__":E$="&&"
6190 REM===INIT PAS DE DEPLACEMENT===
6200 DX=2:DY=2
6210 REM====INIT VALEURS LIMITES=====
6220 L1=02:L2=36:L3=02:L4=26
6230 REM=======PARAMETRES JEU========
6240 NN=.10:SN=80:NJ=3:NE=0
6250 REM====EQUIVALENCES TOUCHES=====
6260 A=53:S=54:O=42:K=24:SP=32:ESC=13
6270 REM=======PREPARER ECRAN========
6280 POKE#26A,10
6290 CLS:PAPER4:INK6
6300 PLOT 2,1," SCORE: 000     "+CHR$(6)+
"  SPACE: 80 "+CHR$(6)+"TIR: 3"
6310 X=20:Y=14:XX=X:YY=Y:A$=D$
6320 RETURN
6330 REM============================
6340 REM======== ROUTINE "USR" ======
6350 REM============================
6360 REM
7000 DATA 08,78,20,10,D8,8A,29,38
7010 DATA 4A,4A,4A,48,8A,29,07,AA
7020 DATA 18,A9,FF,2A,CA,10,FC,AA
7030 DATA A9,0E,20,35,F5,68,09,10
7040 DATA 8D,00,03,A9,08,28,2C,00
7050 DATA 03,F0,03,4C,00,DF,4C,FC
7060 DATA DE
7070 REM============================
7080 REM======== DATA DESSINS =======
7090 REM============================
7100 REM
8000 DATA ù,0,0,12,63,63,12,0,0
8010 DATA _,63,63,63,63,63,63,63,63
8020 DATA &,63,0,63,0,63,0,63,0
8030 DATA FIN
8040 REM
8050 REM============================
8060 REM============================
```

La solution

LES CARRÉS INVISIBLES

Monsieur Gilbert FLAHAUT - Cameroun - nous propose ceci :

Dans la revue MICRORIC N° 3, vous avez présenté un programme très intéressant : "Les carrés invisibles". Vous suggériez dans cet article qu'ORIC puisse fournir une solution si nécessaire.

Si mon courrier vous parvient si tard, c'est que je suis résident au Cameroun et, comme vous le savez peut-être, nous avons eu quelques problèmes de courrier ces dernières semaines.

Le programme que je propose me semble assez court.

Il faut modifier les lignes suivantes (ou plutôt les ajouter au programme existant).

Quelques explications :

Lignes 15-16
On crée le tableau 3×3 T$ comprenant le nom des touches.

Ligne 1106
Nouvelle commande "R".

Ligne 1625
Drapeau indiquant que la solution est demandée.

Lignes 4000 à 4020
On forme une croix + unicolore à l'aide des touches W, A, D, X.

Lignes 4030 à 4080
On modifie les carrés de coin (c'est plus compliqué).

Ligne 4090
Rend si nécessaire le grand carré 3×3 unicolore, invisible.

Ligne 4100
Retour à la musique de victoire.

Ligne 5000
Affiche sous le carré le nom de la touche utilisée lors de la solution.

Ligne 3336
A rajouter dans la présentation.

Les lignes 4000 - 4080 peuvent paraître obscures, mais elles évitent d'écrire plusieurs fois des lignes presque identiques. Si l'on n'avait pas fait cela, le programme aurait été plus clair (et sûrement plus rapide), mais aurait eu une longueur au moins triple de celui proposé.

```
 15 DIMT$(3,3):T1$="QWEASDZXC"
 16 FORI=1TO3:FORJ=1TO3:T$(I,J)=MID$(T1$
,3*(I-1)+J,1):NEXTJ,I
 1106 IFK$="R"THENDR=1:GOTO4000
 1625 IFDR=1THENRETURN
 3336 L=5:GOSUB2:PRINTSPC(4)"POUR LA SOL
UTION, TAPER 'R'":WAIT 100
 4000 FORG=1TO3:FORH=1TO3:IFABS(G-H)<>1T
HEN4020
 4010 IFA(H,G)<>A(2,2)THENK$=T$(G,H):GOS
UB5000:GOSUB1010:GOTO4010
 4020 NEXTH,G
 4030 FORG=1TO3:FORH=1TO3:IFABS(G-H)<>0T
HEN4080
 4040 IFA(H,G)=A(2,2)THEN4080
 4050 FORF=1TO3:U=5+(G=H)+H+(F-1)*(-1_^(
G=H)
 4060 U=U-3*INT(U/3):U=U-3*(U=0)
 4070 K$=T$(F,U):GOSUB5000:GOSUB1010:NEX
TF:GOTO4040
 4080 NEXTH,G
 4090 IFA(2,2)<>PATHENK$="S":GOSUB5000:G
OSUB1010:GOTO4090
 4100 DR=0:GOTO1640
 5000 FOTII=0TO1:PLOT18,23+II,10:PLOT19,
23+II,K$:NEXTII:RETURN
```

Pour adapter le programme "Les carrés invisibles" sur ATMOS il suffit de tenir compte des différences d'affichage avec l'ORIC-1.

Pour PLOT il faut ajouter 1 au premier paramètre qui indique la colonne, ainsi ligne 211Ø remplacer :
PLOT XX, YY+I, CA$ par
PLOT XX+1, YY+I, CA$.

Voir lignes 1ØØ, 2Ø8Ø également et aussi 171Ø, 172Ø, 3Ø7Ø éventuellement.

En ligne 321Ø il est bon de supprimer un PRINT pour éviter le scrolling.

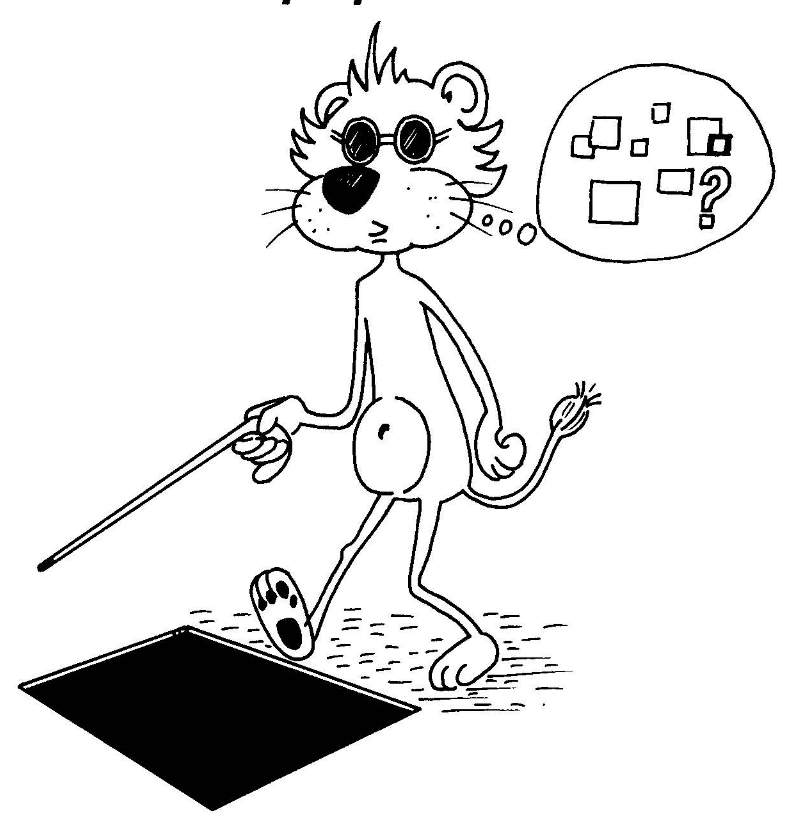
En outre, ligne 5ØØØ on peut écrire avant RETURN LPRINT K$; ce qui fournit la solution, parfois longue sur papier.

Attention en ligne 1 au CALL # F89B qui devient sur ATMOS CALL # F8DØ (reconfiguration des caractères).

Je pense que les lecteurs de MICR'ORIC seront intéressés par ce programme.

CARRÉS INVISIBLES

MICR'ORIC propose une solution :



Il faut d'abord rejeter l'idée de faire essayer par ORIC toutes les combinaisons de touches possibles jusqu'à obtenir la disparition de toutes les cases. En effet, le nombre de combinaisons possibles est très grand : pour un carré de 3×3, il est de n^9, n étant le nombre de couleurs utilisées.

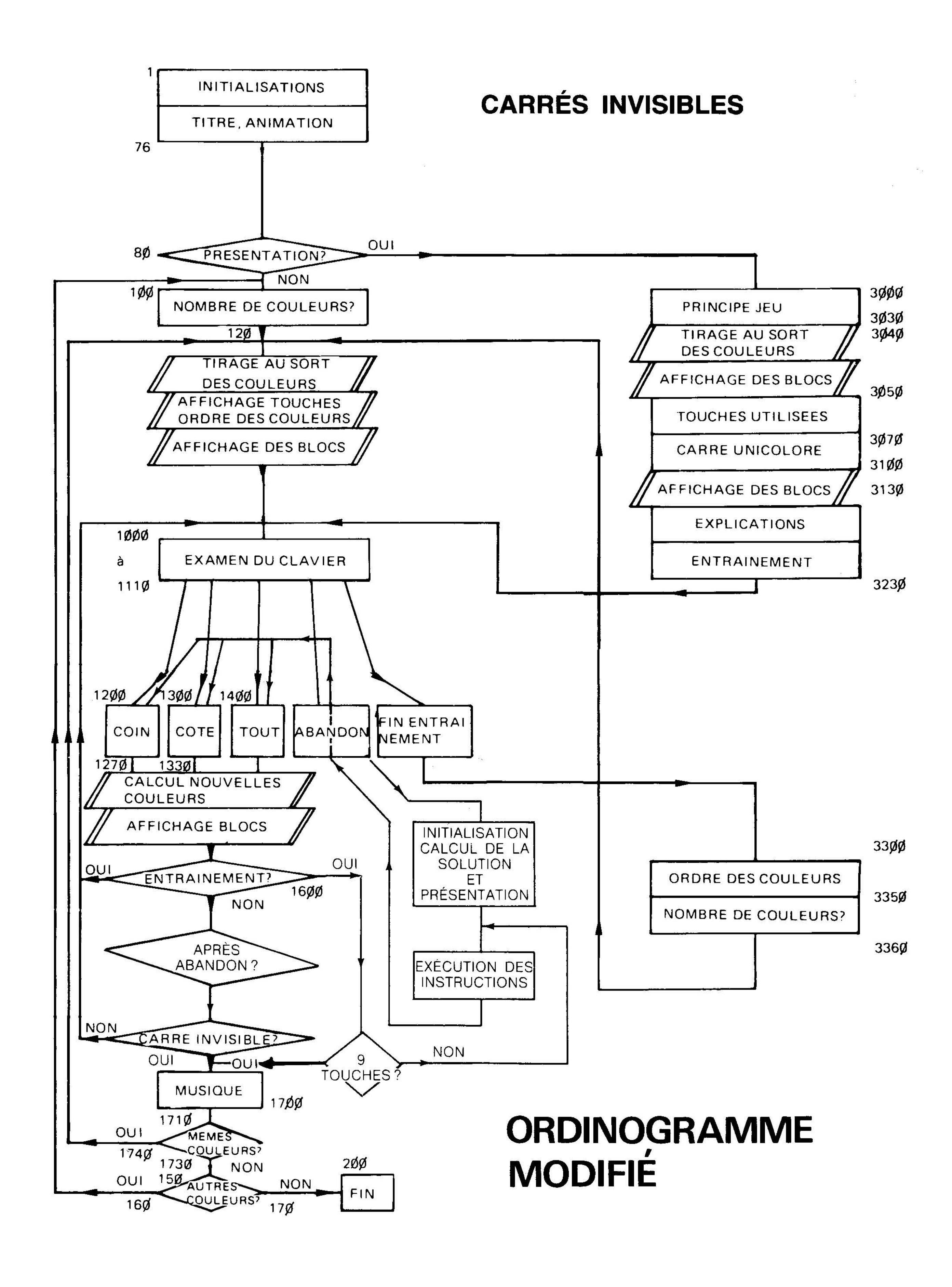
Même en remarquant que le problème est résolu lorsque l'on a obtenu un carré unicolore (il suffit alors d'appuyer sur S autant de foi que nécessaire), et qu'alors le nombre de combinaisons n'est "que" de n^8, on trouve :

Nombre de couleurs	Nombre de combinaisons	Nombre de couleurs	Nombre de combinaisons
2	256	**6**	1 679 616
3	6561	**7**	5 764 801
4	65 536	**8**	16 777 216
5	390 625		

Renonçons donc à une méthode facile à imaginer, mais peu brillante à l'exécution et cherchons autre chose.

Il y a une solution rigoureuse et élégante au problème. Avec un peu de pratique et de réflexion, on peut trouver des algorithmes qui permettent d'arriver à la disparition des cases après un nombre de pressions très limité. L'un d'eux est détaillé.

CARRÉS INVISIBLES



ORDINOGRAMME MODIFIÉ

Ajouter les lignes suivantes au programme publié pages 21-22 de MICR' ORIC n° 3 :

```
15 TC$(1)="S":TC$(2)="W":TC$(3)="A":TC$(
4)="D":TC$(5)="X"

16 TC$(6)="Q":TC$(7)="E":TC$(8)="Z":TC$(
9)="C"

5000 FORI=1TO9:T(I)=0:NEXT:T=0:F=1
5010 T(1)=PA-2*(A(2,1)+A(1,2)+A(3,2)+A(2
,3))+A(1,1)+A(3,1)+A(1,3)+A
(3,3)
5015 T(1)=T(1)+3*A(2,2)
5020 T(6)=A(3,2)-A(3,3)-A(2,2)+A(2,3)
5030 T(7)=A(1,2)-A(2,2)-A(1,3)+A(2,3)
5040 T(8)=A(3,2)-A(2,2)-A(3,1)+A(2,1)
5050 T(9)=A(2,1)-A(2,2)-A(1,1)+A(1,2)
5060 T(2)=A(1,2)-A(1,1)+T(8)
5070 T(3)=A(2,1)-A(1,1)+T(7)
5080 T(4)=A(2,1)-A(3,1)+T(6)
5090 T(5)=A(1,2)-A(1,3)+T(6)
5100 FORI=1TO9
5110 IFT(I)<0THENREPEAT:T(I)=T(I)+CO:UNT
ILT(I)>=0:GOTO5130
5120 IFT(I)>=COTHENREPEAT:T(I)=T(I)-CO:U
NTILT(I)<CO
5130 T=T+T(I):NEXT
5140 PLOT8,23,"ORIC REUSSIT EN        COUP
S"
5145 T$=STR$(T):T$=RIGHT$(T$,LEN(T$)-1):
PLOT24,23,T$
5150 PLOT8,25,"APPUYEZ      FOIS SUR"
5155 FORII=1TO9:IFT(II)=0THEN5200
5160 REPEAT:PLOT29,25,TC$(II)
5165 T$=STR$(T(II)):T$=RIGHT$(T$,LEN(T$)
-1):PLOT17,25,T$
5170 GETK$:IFK$<>TC$(II)THENPING:GOTO517
0
5180 ONIIGOTO1400,1300,1310,1320,1330,12
00,1210,1220,1230
5190 T(II)=T(II)-1:UNTILT(II)=0
5200 NEXTII
5210 F=0:GOTO1640
```

Modifier les lignes suivantes :

```
1100 IFK$="F"THENIFV=0THEN5000

1600 IFF=1THEN5190ELSEES=0:FORX=1TON:FOR
Y=1TON

2030 PLOT2,23,"QWE":PLOT2,24,"ASDF":PLOT
2,25,"ZXC"
```

Dès lors, la modification du programme est simple (voir l'ordinogramme modifié) :

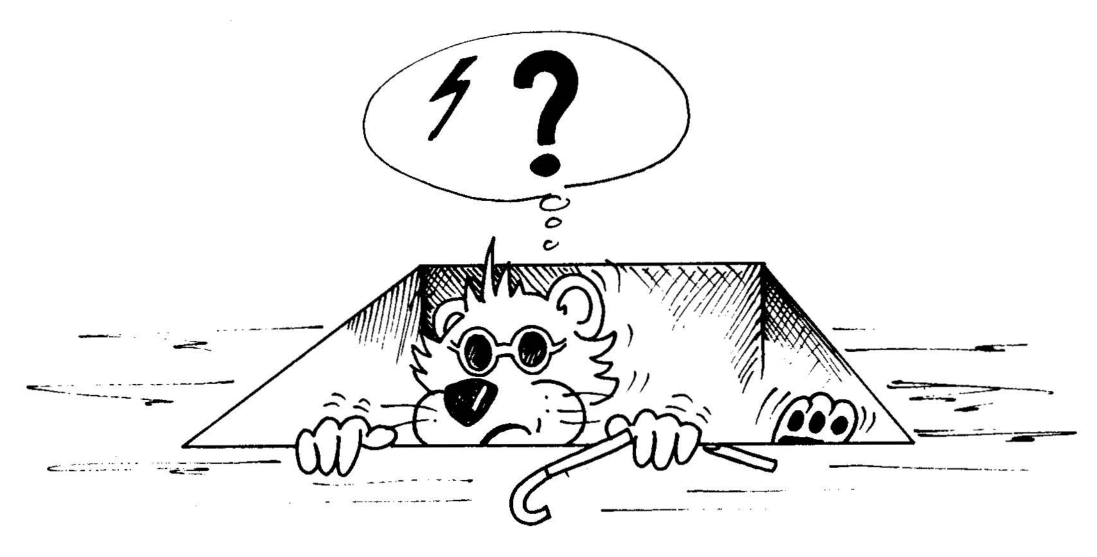
Le programme de calcul et de présentation de la solution est implanté entre 5000 et 5210. Il utilise les caractères TC $(I) qui sont définis en 15 et 16.

Les branchements appropriés sont obtenus par modification des lignes 1100 et 1600.

En reprenant la ligne 2030, on signale la possibilité d'utiliser la touche F.

Le principe du jeu est bien sûr inchangé. Mais si vous "séchez", appuyer sur F : ORIC fournit sa réponse immédiatement, qu'il ne vous reste plus qu'à vérifier en exécutant les consignes affichées en bas de l'écran.

A. de GUERRA



Exemple de solution

Dans les schémas suivants, les cases contenant le même type de hachures sont de la même couleur. Ainsi la première ligne s'interprète de la façon suivante : "appuyer sur W autant de fois que nécessaire pour que les cases hachurées soient de la même couleur". La couleur des cases non hachurées est sans importance.

Numéro d'ordre	Appuyer sur	Pour obtenir
1	W	
2	X	
3	C	
4	E	
5	D	
6	A	

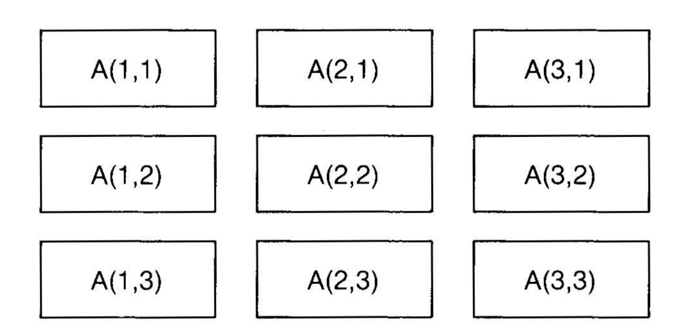
Numéro d'ordre	Appuyer sur	Pour obtenir
7	C	
8	X	
9	Q	
10	D	
11	W	
12	Z	

13	**APPUYER SUR S !**

La traduction mathématique de cet enchaînement n'est pas difficile ; elle demande surtout beaucoup de précision et de rigueur : il faut prendre en compte le changement de couleur de toutes les cases impliquées par la pression d'une touche. On obtient finalement 9 égalités qui déterminent le nombre de pressions à effectuer sur les 9 touches du jeu.

Calcul de la solution

La couleur de la case de coordonnées X, Y est repérée par A(X,Y). Au départ, le carré, défini par tirage aléatoire, a pour représentation :



Par exemple, si A(1,1) = 1, la case supérieure gauche est rouge. En appuyant une fois sur W, les trois cases du haut progressent d'une unité dans l'ordre des couleurs ; dans notre exemple, A(1,1) prend la valeur Z, et la case devient verte. "L'écart" entre deux couleurs est donné par la différence de leurs codes : l'écart entre bleu et rouge est de 3, et il faut appuyer 3 fois sur la (ou les) touche(s) concernée(s) pour qu'une case rouge devienne bleue. Avec nos notations, l'écart de couleur entre les cases (1,1) et (1,2) est donc A(1,2) — A(1,1).

L'écriture de la première opération de l'encadré 1 se fait de la façon suivante :

Pour rendre les deux cases hachurées de la même couleur, il faut appuyer [A(1,2) — A(1,1)] fois sur la touche W (remarquons qu'il n'y a pas d'inconvénient à obtenir un nombre négatif puisque, dans notre programme, les codes de couleur sont cycliques : si l'on joue avec 8 couleurs, le rouge est représenté par les nombres..., —15, —7, 1, 9, 17,.... La correction indispensable pour qu'ORIC interprète ce nombre comme une couleur d'encre est faite après calcul, aux lignes 5100 à 5130). Après cette opération, la représentation du carré est :

A(1,2)	A(2,1) + A(1,2) — A(1,1)	A(3,1) + A(1,2) — A(1,1)
A(1,2)	A(2,2)	A(3,2)
A(1,3)	A(2,3)	A(3,3)

De même, pour réaliser l'étape 2, il faut appuyer [A(1,2) — A(1,3)] fois sur X ; ceci affecte d'autant les cases (2,3) et (3,3), et ainsi de suite. Les expressions prennent vite une longueur considérable, mais, en restant calme, l'on aboutit aux relations suivantes :

Nombre de pressions sur S (PA=couleur du fond)

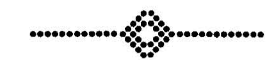
TS = PA — 2.[A(2,1) + A(1,2) + A(3,2) + A(2,3)] + [A(1,1) + A(3,1) + A(1,3) + A(3,3) + 3.A(2,2)]

Sur Q : TQ = A(3,2) — A(3,3) — A(2,2) + A(2,3)
Sur E : TE = A(1,2) — A(2,2) — A(1,3) + A(2,3)
Sur Z : TZ = A(3,2) — A(3,1) — A(2,2) + A(2,1)
Sur C : TC = A(2,1) — A(1,1) — A(2,2) + A(1,2)
Sur W : TA = A(1,2) — A(1,1) + TZ
Sur A : TA = A(2,1) — A(1,1) + TE
Sur D : TD = A(2,1) — A(3,1) + TQ
Sur X : TX = A(1,2) — A(1,3) + TQ

Ces expressions sont calculées aux lignes 5010 à 5090.

(suite de la page 61)

```
4070 REM
4080 REM
4100 ADD=#8000-(EC-1)*#501
4110 DOKE#F8,ADD:DOKE#FA,#BBA8
4120 CALL #9505:GOSUB 5000
4130 GOTO 450
5000 L=L+1:IF L=7 THEN L=1
5005 IF L=4 THEN L=5
5010 FOR X=#BBA8 TO #BFB8 STEP
#28:POKE X,L:NEXT
5020 DOKE #12,#BB93::"Temps :
0 00 000 —"
5030 C=INT(TEMP/100):A=INT(TEM
P/10)-C*10:B=TEMP-A*10-C*100+48
:A=A+48:C=C+48
5040 POKE #BB9D,C:POKE #BB9F,A
5050 POKE #BBA0,B:DOKE #BB80,#
0C03:POKE #BB92,5:POKE #BB8A,2
5060 RETURN
6000 REM
6010 B=VAL(A$)
6020 PLAY3,0,1,1000:M1=INT((B/
9)*4+2)
6030 FOR M0=1 TO 12
6040 MUSIC 1,M1,M0,0
6050 NEXT M0
6060 RETURN
```

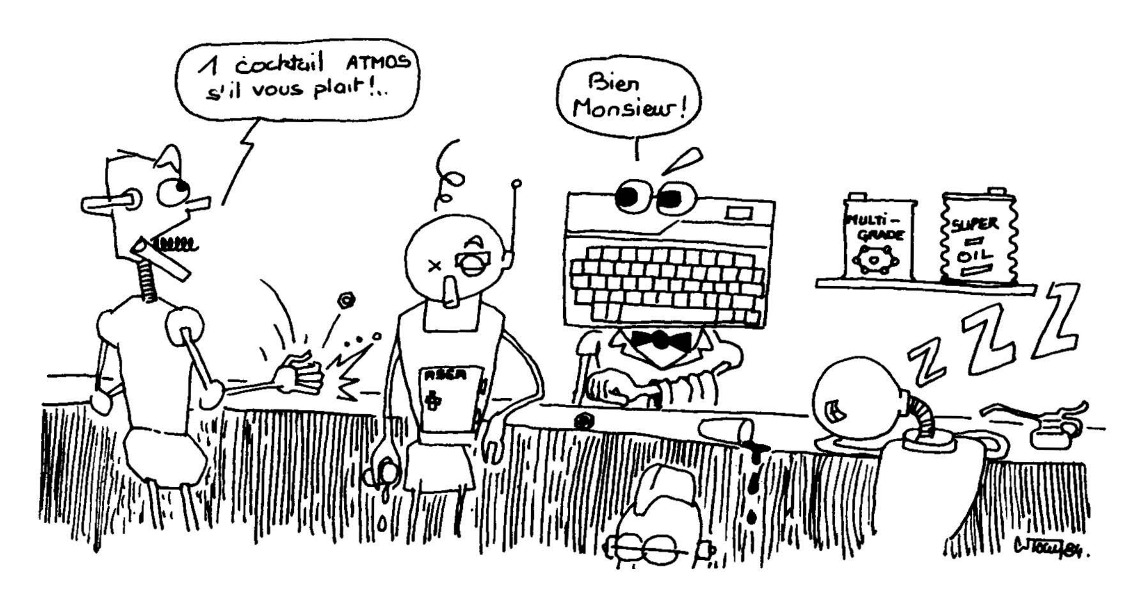




COCKTAILS

par Philippe BRAX

Avez-vous déjà essayé de confectionner un cocktail en cherchant une recette sur un livre ?



Vous êtes obligé de passer un grand nombre de recettes en revue car il vous manque souvent un des produits nécessaires à la confection du cocktail. D'où l'idée d'informatiser la recherche, qui se fera en fonction des seuls produits dont vous disposez.

Le programme vous propose d'autres services :

- La liste des produits de votre bar.
- La liste des cocktails en fonction d'autres produits que vous désireriez utiliser.
- La recette d'un cocktail à partir de son nom.
- La liste des cocktails contenant un ou plusieurs produits. (par exemple : liste des cocktails contenant du whisky).

Le programme contient 115 recettes de cocktails Vous pourrez en ajouter d'autres mais attention à la capacité mémoire. En supprimant les dessins HIRES et en utilisant GRAB vous aurez plus de place.

Chaque cocktail est inscrit de la manière suivante en DATA :

— Nombre de produits, nom de chaque produit, nom du cocktail.

— Recette proprement dite sur deux DATA successifs.

Vous devrez également remplacer les produits en DATA par ceux dont vous disposez et en tenir la liste à jour ou reconstituer votre stock de bouteilles.

Le programme a été écrit pour ORIC-1. Pour l'adapter à l'ATMOS il suffit en ligne 130 de rectifier l'affichage du titre et de modifier les TAB.

```
10 REM *************************
20 REM ****                 ****
30 REM ****    COCKTAILS    ****
40 REM ****                 ****
50 REM *************************
60 REM
70 CLS:PAPER0:INK3:POKE618,10
75 PLOT5,12,"15 secondes de patience..."
80 GOSUB5000
85 AT$="APPUYEZ SUR UNE TOUCHE"
90 DIMA$(99):DIMB$(150):DIMA(150):DIMC$(7,150):DIMD$(150):DIME$(150)
95 DIMF$(150)
110 GOSUB3000:REM INITIALISATION
120 CLS:PAPER3:INK4:PRINT
130 PRINTCHR$(4);CHR$(27);CHR$(84);CHR$(27)"J         C O C K T A I L S":PRINTCHR$(4)
140 PRINT:PRINT
150 PRINT:PRINT"1:Liste des produits du bar"
160 PRINT:PRINT"2:Liste des cocktails possibles en      fonction du bar"
170 PRINT:PRINT"3:Cocktails possibles en fonction de    produits de votre choix"
180 PRINT:PRINT"4:Recettes de cocktails"
190 PRINT:PRINT"5:Cocktails contenant 1 (ou n)          produit(s).choisi(s)
200 PRINT:PRINT"6:Quelques explications"
210 PRINT:PRINT"7:Fin"
220 PLOT3,25,"SELECTIONNEZ UN NUMERO"
230 PRINT:PRINT"J'ai ";K-1;"cocktails en m[moire"
240 GETA$:A=VAL(A$):IFA<1ORA>7THENPING:GOTO120
250 ON A GOTO 300,400,800,1200,1400,1800,2500
300 REM
310 REM * LISTE DES PRODUITS DU BAR *
320 REM
325 CLS:PRINT:PAPER0:INK3
330 PRINTCHR$(27);CHR$(65);CHR$(4);CHR$(27)"J  Liste des produits du bar";CHR$(4)
335 PRINT:PRINT:PRINT
340 FORA=1TOI:PRINTB$(A)SPC(2):IF POS(0)>25THENPRINT
350 NEXT:PRINT:PRINT:PRINT
360 PRINT"Appuyez sur une touche"
370 GETZ$:CLS:GOTO120
400 REM
410 REM * LISTE DES COCKTAILS *
411 REM *     EN FONCTION     *
412 REM *       DU BAR        *
420 REM
425 PAPER0:HIRES:POKE618,10
430 FORZ=0TO198STEP2
440 CURSET0,Z,0:FILL1,1,17:CURSET0,199-Z,0:FILL1,1,20
445 NEXT
450 M$="COCKTAILS REALISABLES":N$="AVEC LES PRODUITS DE VOTRE BAR"
460 INK3
470 CURSET50,70,0:FORN=1TOLEN(M$):CURMOV6,0,0
480 CHARASC(MID$(M$,N,1)),0,1:NEXT
490 CURSET30,90,0:FORN=1TOLEN(N$):CURMOV6,0,0
500 CHARASC(MID$(N$,N,1)),0,1:NEXT
510 WAIT300:TEXT:POKE618,10:INK6
520 PRINT:PRINT:PRINT" Je cherche..."
530 K=0:G=0:NC=0
540 REPEAT
545 :  K=K+1:N=0
550 :  FORD=1TOA(K)
560 :    FORB=1TOI
570 :      IFB$(B)=C$(D,K)THENN=N+1
580 :      IFN=A(K)THENGOSUB700:                   GOTO545
590 :    NEXTB
600 :  NEXTD
610 UNTILD$(K)="C'est tout..."
620 IF G<>0THENPRINTD$(K)ELSEPRINT"Je ne peux rien vous proposer !"
```

```
630 PRINT"Appuyez sur une touche"
640 GETZ$:GOTO120
700 PRINT:PRINT" "CHR$(27)"A";D$(K)
710 PRINTE$(K):PRINTF$(K):PRINT
720 G=G+1:NC=NC+1:IFNC=3THENPRINT" ";CHR$(27);"B";AT$:GETZ$:NC=0
730 RETURN
795 REM
800 REM  *   COCKTAILS REALISABLES  *
810 REM  * EN FONCTION DE  PRODUITS *
820 REM  *      DE VOTRE CHOIX      *
825 REM
827 PAPER0
830 CLS:FORI=1TO26:PLOT0,I,16+INT(RND(1)*7):NEXT
840 WAIT100:CLS:PAPER6:INK0:PRINT:PRINT
850 PRINT"Tapez sur le clavier les produits":PRINT:PRINT"dont vous disposez."
860 PRINT:PRINT"Appuyez sur la touche RETURN apr]s":PRINT:PRINT"chaque produit"
870 PRINT:PRINT"Appuyer sur F lorsque vous avez":PRINT:PRINT"termin[."
880 PRINT:PRINT:PRINT"Appuyez sur une touche.":GETZ$:PRINTCHR$(17)
890 CLS:PAPER4:INK7:PRINT:II=0
900 REPEAT
910 : II=II+1
920 : PRINT"Produit no ";II:PRINTTAB(25);:INPUTAA$(II)
930 UNTILAA$(II)="F":II=II-1
940 IFII=1THENPRINT:PRINT"Un cocktail necessite plusieurs pro-  duits"ELSE945
942 WAIT300:GOTO890
945 CLS:PRINT:PRINTCHR$(17):PRINT"Je cherche..."
950 K=0:G=0:NC=0
960 REPEAT
970 :   K=K+1:N=0
980 :   FORD=1TOA(K)
990 :     FORB=1TOII
1000 :      IFAA$(B)=C$(D,K)THENN=N+1
1010 :      IFN=A(K)THENGOSUB1080          :      GOTO970
1020 :    NEXTB
1030 :  NEXTD
1040 UNTILD$(K)="C'est tout..."
1050 IFG<>0THENPRINT:PRINTD$(K)ELSEPRINT:PRINT"Je ne peux rien vous proposer !"
1060 PRINT:PRINTAT$
1070 GETZ$:II=0:GOTO120
1080 PRINT:PRINT" ";CHR$(27);"A";D$(K)
1090 PRINTE$(K):PRINTF$(K):PRINT
1100 G=G+1:NC=NC+1:IFNC=3THENPRINT:PRINT" ";CHR$(27);"B";AT$:GETZ$:NC=0
1110 RETURN
1200 REM
1210 REM   *  RECETTES  *
1220 REM
1225 PAPER0:INK6:HIRES:POKE 618,10
1230 FILL200,40,22:CURSET16,13,0:FILL176,35,19
1240 CURSET32,26,0:FILL150,29,17:CURSET48,39,0:FILL124,24,18
1250 CURSET64,52,0:FILL98,19,16:CURSET80,65,0:FILL72,13,20
1255 CURSET92,78,0:FILL46,8,21
1260 WAIT200:TEXT:PAPER6:INK4
1270 PRINT:PRINT"Indiquez moi le nom d'un cocktail,    je vous en donne la recette.":PRINT
1280 PRINT:INPUT"Quel est le nom du cocktail              ";Z$:PRINT:PRINT
1290 FORZ=1TOK
1300 IFZ$=D$(Z)THEN1330
1310 NEXT
1320 PRINT:PRINT:PRINT"Je ne connais pas ce cocktail,d[sol[...":GOTO1340
1330 PRINT:PRINT"Ce cocktail se compose de :":PRINT:PRINTE$(Z):PRINTF$(Z)
1340 PRINT:PRINT:PRINT"D[sirez vous une autre recette ?(O.N)":GETZ$
1350 IFZ$="O"THENCLS:GOTO1280ELSEPOKE618,10:GOTO120
1400 REM
1410 REM  *    COCKTAILS     *
1420 REM  *   CONTENANT X    *
1430 REM  * PRODUITS CHOISIS *
```

```
1440 REM
1450 CLS:PAPER0:INK2
1460 PRINT"Cette option vous permet d'avoir la   liste et la recette de cocktails"
1470 PRINT"contenant 1 ou plusieurs produits que vous aurez choisi"
1480 PRINT:PRINT"Exemple:liste des cocktails contenant du Champagne"
1490 PRINT:PRINT"Tapez le nom du ou des produits que   vous souhaitez voir utilis[s."
1495 PRINT:PRINT"Lorsque vous avez termin[,appuyez sur F puis sur RETURN."
1500 PRINT:PRINT:PRINTAT$
1515 GETZ$:CLS:PAPER3:INK1:II=0:NC=0:PRINTCHR$(17)
1520 REPEAT
1530 :  II=II+1
1540 :  PRINT"Produit no ";II
1550 :  PRINTTAB(25):INPUTAA$(II)
1560 UNTILAA$(II)="F"
1565 II=II-1:K=0:PRINTCHR$(17):PRINT:PRINT"Je cherche..."
1570 REPEAT
1580 :  K=K+1:N=0
1590 :  FORD=1TOA(K)
1600 :    FORB=1TOII
1610 :      IFAA$(B)=C$(D,K)THENGOSUB              1700
1620 :    NEXTB
1630 :  NEXTD
1640 UNTILD$(K)="C'est tout..."
1650 PRINT:PRINTD$(K):GOTO1720
1700 PRINT:PRINT" "CHR$(27)"E";D$(K):PRINTE$(K):PRINTF$(K)
1705 NC=NC+1:IFNC=3THENPRINT:PRINT" ";CHR$(27);"D";AT$:GETZ$:NC=0
1710 RETURN
1720 PRINT:PRINT:PRINTAT$
1730 GETZ$:CLS:PRINT:PRINT"Voulez vous essayer avec d'autres     produits ?"
1740 GETZ$:IFZ$="O"THENCLS:PRINTCHR$(17):II=0:GOTO1520ELSE120
1790 REM
1800 REM     *   MODE D'EMPLOI  *
1810 REM     *   DU PROGRAMME   *
1820 REM
1830 CLS:PAPER0:INK6:POKE618,10
1930 PRINT"Chaque recette est pr[sent[e de la "
1940 PRINT"de la mani]re suivante :":PRINT
1950 PRINT"nom du cocktail,suivi sur la 2[me     ligne d'un ou deux symboles,"
1960 PRINT"puis de la recette proprement dite":PRINT
1970 PRINT"Les symboles sont les suivants :":PRINT
1980 PRINT"% : Shaker"
1990 PRINT"> : Verre @ m[lange":PRINT
2000 PRINT"En l'absence d'un de ces symboles,le  cocktail se pr[pare directement"
2010 PRINT"dans le verre de service.":PRINT
2020 PRINT"< : verre gobelet (25 ou 33 cl)"
2030 PRINT"} : verre @ whisky (25 cl)"
2040 PRINT"_ : verre @ pied (12 cl)"
2050 PRINT"\ : verre @ cocktail (6 cl)
2060 PRINT"& : verre tulipe ou @ champagne"
2080 PRINT:PRINT:PRINT"Le verre doseur utilis[ dans les      recettes,contient 4 cl."
2090 GETZ$:GOTO120
2500 CLS:PLOT15,15,"AU REVOIR"
2510 WAIT300:PRINTCHR$(20):PAPER0:INK6:END
3000 REM
3010 REM *** INITIALISATION ***
3020 REM
3025 REM --- PRODUITS DU BAR ---
3030 K=0:I=0
3040 REPEAT
3050 : I=I+1:READB$(I)
3060 UNTIL B$(I)="Z"
3065 I=I-1
3070 REM --- COCKTAILS ---
3080 REPEAT
```

(remarque : — correspond à £)

```
3090 :  K=K+1:READA(K)
3100 :  FORJ=1 TO A(K)
3110 :    READC$(J,K)
3120 :  NEXT
3130 :  READD$(K),E$(K),F$(K)
3140 UNTILD$(K)="C'est tout..."
3150 RETURN
5000 REM ** ACCENTUATION **
5010 REM
5020 FORI=1TO13:READC$:C=ASC(C$)
5030 FORN=0TO7:READB
5040 POKE46080+8*C+N,B
5050 NEXTN,I
5060 RETURN
6000 DATA@,8,4,28,2,30,34,30,0
6010 DATA^,0,0,14,16,16,14,4,8
6020 DATA{,28,34,28,34,62,32,30,0
6030 DATA[,4,8,28,34,62,32,30,0
6040 DATA],16,8,28,34,62,32,30,0
6060 DATA£,20,0,24,8,8,8,28,0
6100 DATA\,0,0,34,20,8,8,28,0
6110 DATA_,34,20,20,8,8,8,28,0
6120 DATA&,20,34,34,20,8,8,28,0
6130 DATA<,34,34,34,34,34,34,62,0
6140 DATA>,2,4,42,50,50,50,62,0
6160 DATA%,12,18,30,18,18,18,30,0
6170 DATA},0,0,34,34,34,34,62,0
6985 REM
6990 REM ** LISTE DES PRODUITS **
6991 REM **       DU BAR         **
6992 REM
7000 DATA "ANGOSTURA","ALCOOL DE PRUNE","ARMAGNAC"
7010 DATA"BOURBON","VIN BLANC SEC","VIN BLANC DOUX","BANANE"
7020 DATA"BIERE BLONDE"
7030 DATA"CALVADOS","CREME FRAICHE","CHAMPAGNE","CAFE","CURACAO BLEU","COGNAC"
7040 DATA"COINTREAU"
7045 DATA"EAU GAZEUSE"
7050 DATA"GIN"
7060 DATA"IZARRA VERTE"
7070 DATA"JUS D'ORANGE","JUS DE CITRON","JUS DE PAMPLEMOUSSE"
7080 DATA"KIRSCH"
7090 DATA"LIMONADE","LIQUEUR D'ABRICOT","LAIT","LIQUEUR DE CHATAIGNE"
7100 DATA"LIQUEUR DE POIRE"
7110 DATA"MARTINI BLANC"
7120 DATA"PERNOD"
7130 DATA"OEUF"
7140 DATA"RHUM BLANC","RHUM BRUN"
7150 DATA"VERMOUTH","VODKA"
7160 DATA"WHISKY"
7300 DATA"Z"
7400 DATA3,"GIN","EAU GAZEUSE","JUS DE CITRON","GIN FIZZ"
7410 DATA"%< 1 CUILLERE A CAFE DE SUCRE,1 MESUREDE GIN,LE JUS D'1 CITRON"
7420 DATA"REMPLIR D'EAU GAZEUSE,GLACONS DANS LE SHAKER."
7440 DATA2,"PORTO","VODKA","BOMBARRAL"
7441 DATA"%& METTRE DANS LE SHAKER DES CUBES DE GLACE,1/3 DE VODKA,2/3 DE "
7442 DATA"PORTO.FIXER AU BORD DU VERRE UN ZESTE DE CITRON"
7450 DATA2,"PORTO","GIN","BERLENGA"
7451 DATA"%& METTRE DANS LE SHAKER DE LA GLACE, 2/3 DE PORTO,1/3 DE GIN."
7452 DATA"FRAPPER ET PASSER DANS LE VERRE,FIXER UNE TR. DE CITRON."
7460 DATA3,"ANGOSTURA","VERMOUTH","XERES","ADONIS"
7461 DATA"<  GLACE,1/3 DE VERMOUTH,2/3 DE XERES 1 TRAIT D'ANGOSTURA"," "
7470 DATA2,"XERES","JUS D'ORANGE","ANDALUZ"
7471 DATA"%< GLACE,1 CUILLERE A SOUPE DE XERES, 2/3 DE VER.DE JUS D'ORANGE"
7472 DATA""
```

Suite de ce PROGRAMME
dans notre prochain numéro

N.B. : En fin de DATA mettre "C'est tout..."

B.D.

CAPTAIN TANEX & ORIC-Ω

TROP TARD POUR CHERCHER AILLEURS DANS LES CATACOMBES

CES BOTTES DOIVENT ÊTRE MAGIQUES

ENFILONS-LES ÇA POURRAIT SERVIR

JE TE TIENS MAINTENANT

CAPTAIN TANEX !

L'ŒIL DU MANDAR

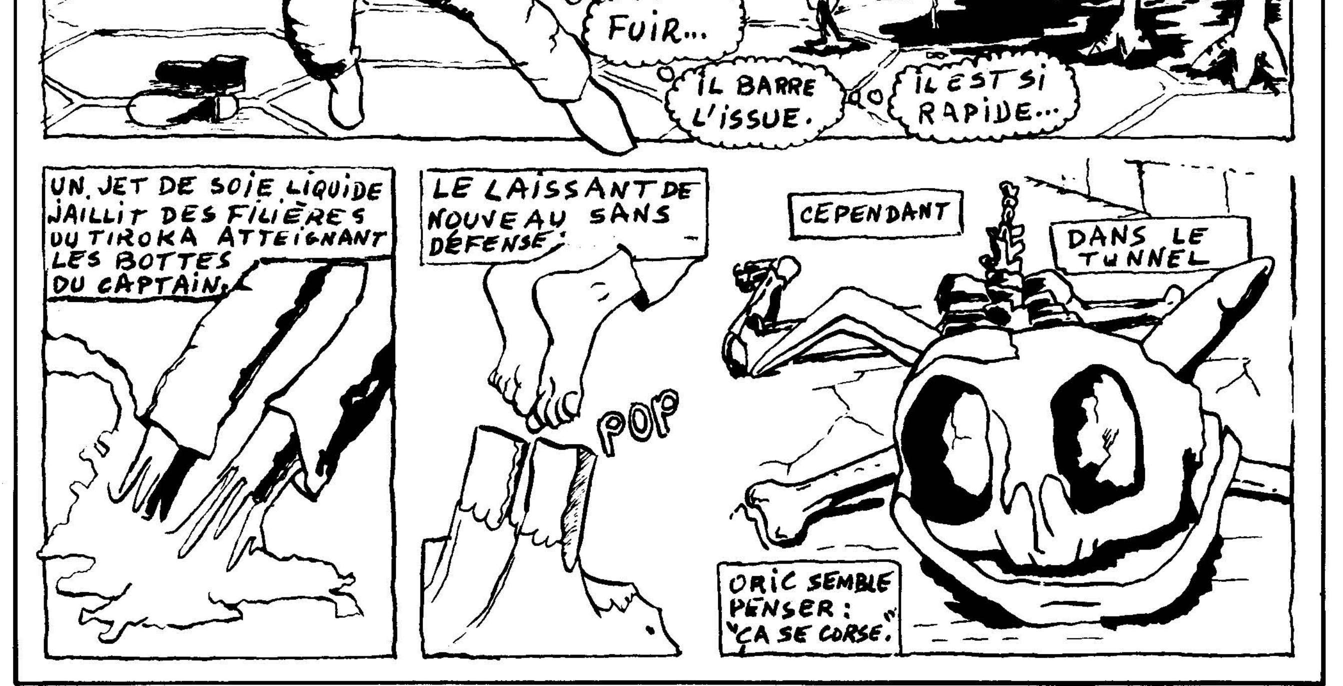
SWISH

QUOI!

FUIR...

IL BARRE L'ISSUE.

IL EST SI RAPIDE...



LES ADRESSES DES FONCTIONS

Pour vous éviter de chercher vous-même les adresses des routines selon la méthode proposée par Fabrice Broche dans ce numéro, nous vous proposons des tableaux imprimés, disposés en pages centrales, détachables et d'un emploi commode.

Nous avons trié par CODE, par ADRESSE et par FONCTION en séparant V1.Ø de V1.1. Toutes les valeurs sont héxadécimales. Celles indiquées entre parenthèses ne sont pas directement accessibles.

Pour tout savoir à propos d'une instruction BASIC particulière il vous faudra désassembler la ROM à partir de l'adresse correspondante.

Ensuite, essayer de comprendre !! Les ruses et les pièges sont nombreux...mais, avec de la persévérance, on peut y arriver.

Dans les prochains numéros de MICR'ORIC nous essayerons de faire ce travail délicat avec vous et d'en tirer le meilleur profit possible pour vos propres programmes en BASIC ou en langage machine.

Guy JUY & Christian MAGRIN

LISTE DES ADRESSES DES FONCTIONS PAR CODES. V1.0

CODE	FONCTION	ADRESSES
80	END	C941
81	EDIT	C6A5
82	INVERSE	CFE4
83	NORMAL	CFE4
84	TRON	CC8C
85	TROFF	CC8F
86	POP	C9E0
87	PLOT	D9C6
88	PULL	DA16
89	LORES	D937
8A	DOKE	D8AC
8B	REPEAT	D9FA
8C	UNTIL	DA16
8D	FOR	C841
8E	LLIST	C824
8F	LPRINT	C832
90	NEXT	CE0C
91	DATA	CA09
92	INPUT	CCC9
93	DIM	D0F2
94	CLS	CC0A
95	READ	CCFD
96	LET	CAD2
97	GOTO	C9B3
98	RUN	C990
99	IF	CA3E
9A	RESTORE	C91F
9B	GOSUB	C996
9C	RETURN	C9E0
9D	REM	CA61
9E	HIMEM	E95B
9F	GRAB	E974
A0	RELEASE	E994
A1	TEXT	F923
A2	HIRES	FBE3
A3	SHOOT	FA9B
A4	EXPLODE	FAB1
A5	ZAP	FAC7
A6	PING	FA85
A7	SOUND	FB26
A8	MUSIC	FBFE
A9	PLAY	FBB6
AA	CURSET	F02D
AB	CURMOV	F064
AC	DRAW	F079
AD	CIRCLE	F2E5
AE	PATTERN	F093
AF	FILL	F1E5
B0	CHAR	F0A5
B1	PAPER	F17F
B2	INK	F18B
B3	STOP	C93F
B4	ON	CA78
B5	WAIT	D89D
B6	CLOAD	E7BA
B7	CSAVE	E7DB
B8	DEF	D401
B9	POKE	D894
BA	PRINT	CB61
BB	CONT	C970

CODE	FONCTION	ADRESSES
BC	LIST	C773
BD	CLEAR	C73A
BE	GET	CCBA
BF	CALL	E80D
C0	!	CC89
C1	NEW	C71B
C2	TAB(	(CBCA)
C3	TO	(C86C)
C4	FN	(D454)
C5	SPC(	(CBCA)
C6	à	----
C7	AUTO	(E76B)
C8	ELSE	(CA6D)
C9	THEN	(CA48)
CA	NOT	CFB0
CB	STEP	(C894)
CC	+	DA9A
CD	-	DA83
CE	*	DCBA
CF	/	DDE3
D0	↑	E234
D1	AND	D05A
D2	OR	D057
D3	>	D087
D4	=	D087
D5	<	D087
D6	SGN	DF12
D7	INT	DFA5
D8	ABS	DF31
D9	USR	0021
DA	FRE	D3D6
DB	POS	D3FA
DC	HEX$	D917
DD	&	02FB
DE	SQR	E22A
DF	RND	E34B
E0	LN	DC79
E1	EXP	E2A6
E2	COS	E387
E3	SIN	E38E
E4	TAN	E3D7
E5	ATN	E43B
E6	PEEK	D87D
E7	DEEK	D8C8
E8	LOG	DDD0
E9	LEN	D7EB
EA	STR$	D4D8
EB	VAL	D81C
EC	ASC	D7FA
ED	CHR$	D75B
EE	PI	D8EE
EF	TRUE	DF00
F0	FALSE	DEFC
F1	KEY$	DA4F
F2	SCRN	D9B4
F3	POINT	E9CD
F4	LEFT$	D76F
F5	RIGHT$	D79B
F6	MID$	D7A6
F7	GO	----

LISTE DES ADRESSES DES FONCTIONS PAR CODES. V.1.1.

CODE	FONCTION	ADRESSES
80	END	C973
81	EDIT	C692
82	STORE	E987
83	RECALL	E9D1
84	TRON	CD16
85	TROFF	CD19
86	POP	CA12
87	PLOT	DA51
88	PULL	DAA1
89	LORES	D9DE
8A	DOKE	D967
8B	REPEAT	DA85
8C	UNTIL	DAA1
8D	FOR	C855
8E	LLIST	C7FD
8F	LPRINT	C809
90	NEXT	CE98
91	DATA	CA3C
92	INPUT	CD55
93	DIM	D17E
94	CLS	CCCE
95	READ	CD89
96	LET	CB1C
97	GOTO	C9E5
98	RUN	C9BD
99	IF	CA70
9A	RESTORE	C952
9B	GOSUB	C9C8
9C	RETURN	CA12
9D	REM	CA99
9E	HIMEM	EBCE
9F	GRAB	EBE7
A0	RELEASE	EC0C
A1	TEXT	EC21
A2	HIRES	EC33
A3	SHOOT	FAB5
A4	EXPLODE	FACB
A5	ZAP	FAE1
A6	PING	FA9F
A7	SOUND	FB40
A8	MUSIC	FC18
A9	PLAY	FBD0
AA	CURSET	F0C8
AB	CURMOV	F0FD
AC	DRAW	F110
AD	CIRCLE	F37F
AE	PATTERN	F11D
AF	FILL	F268
B0	CHAR	F12D
B1	PAPER	F204
B2	INK	F210
B3	STOP	C971
B4	ON	CAC2
B5	WAIT	D958
B6	CLOAD	E85B
B7	CSAVE	E909
B8	DEF	D4BA
B9	POKE	D94F
BA	PRINT	CBAB
BB	CONT	C9A0

CODE	FONCTION	ADRESSES
BC	LIST	C748
BD	CLEAR	C70D
BE	GET	CD46
BF	CALL	E946
C0	!	ED13
C1	NEW	C6EE
C2	TAB(	(CC2E)
C3	TO	(C880)
C4	FN	(D50D)
C5	SPC(	(CC2E)
C6	à	(CBC7)
C7	AUTO	(E7FE)
C8	ELSE	(CAA9)
C9	THEN	(CA7C)
CA	NOT	D03C
CB	STEP	(C8A8)
CC	+	DB25
CD	-	DB0E
CE	*	DCF0
CF	/	DDE7
D0	↑	E238
D1	AND	D0E6
D2	OR	D0E3
D3	>	D113
D4	=	D113
D5	<	D113
D6	SGN	DF21
D7	INT	DFBD
D8	ABS	DF49
D9	USR	0021
DA	FRE	D47E
DB	POS	D4A6
DC	HEX$	D9B5
DD	&	02FB
DE	SQR	E22E
DF	RND	E34F
E0	LN	DCAF
E1	EXP	E2AA
E2	COS	E38B
E3	SIN	E392
E4	TAN	E3DB
E5	ATN	E43F
E6	PEEK	D938
E7	DEEK	D983
E8	LOG	DDD4
E9	LEN	D8A6
EA	STR$	D593
EB	VAL	D8D7
EC	ASC	D8B5
ED	CHR$	D816
EE	PI	DE77
EF	TRUE	DF0F
F0	FALSE	DF0B
F1	KEY$	DADA
F2	SCRN	DA3F
F3	POINT	EC45
F4	LEFT$	D82A
F5	RIGHT$	D856
F6	MID$	D861

LISTE DES ADRESSES DES FONCTIONS PAR ADRESSES. V.1.0.

ADRESSES	CODE	FONCTION
----	C6	à
----	F7	GO
0021	D9	USR
02FB	DD	&
C6A5	81	EDIT
C71B	C1	NEW
C73A	BD	CLEAR
C773	BC	LIST
C824	8E	LLIST
C832	8F	LPRINT
C841	8D	FOR
(C86C)	C3	TO
(C894)	CB	STEP
C91F	9A	RESTORE
C93F	B3	STOP
C941	80	END
C970	BB	CONT
C990	98	RUN
C996	9B	GOSUB
C9B3	97	GOTO
C9E0	86	POP
C9E0	9C	RETURN
CA09	91	DATA
CA3E	99	IF
(CA48)	C9	THEN
CA61	9D	REM
(CA6D)	C8	ELSE
CA78	B4	ON
CAD2	96	LET
CB61	BA	PRINT
(CBCA)	C2	TAB(
(CBCA)	C5	SPC(
CC0A	94	CLS
CC89	C0	!
CC8C	84	TRON
CC8F	85	TROFF
CCBA	BE	GET
CCC9	92	INPUT
CCFD	95	READ
CE0C	90	NEXT
CFB0	CA	NOT
CFE4	82	INVERSE
CFE4	83	NORMAL
D05A	D1	AND
D057	D2	OR
D087	D3	>
D087	D4	=
D087	D5	<
D0F2	93	DIM
D3D6	DA	FRE
D3FA	DB	POS
D401	B8	DEF
(D454)	C4	FN
D4D8	EA	STR$
D75B	ED	CHR$
D76F	F4	LEFT$
D79B	F5	RIGHT$
D7A6	F6	MID$
D7EB	E9	LEN
D7FA	EC	ASC
D81C	EB	VAL
D87D	E6	PEEK
D894	B9	POKE
D89D	B5	WAIT
D8AC	8A	DOKE
D8C8	E7	DEEK
D8EE	EE	PI
D917	DC	HEX$
D937	89	LORES
D9B4	F2	SCRN
D9C6	87	PLOT
D9FA	8B	REPEAT
DA16	88	PULL
DA16	8C	UNTIL
DA4F	F1	KEY$
DA83	CD	-
DA9A	CC	+
DC79	E0	LN
DCBA	CE	*
DDD0	E8	LOG
DDE3	CF	/
DEFC	F0	FALSE
DF00	EF	TRUE
DF12	D6	SGN
DF31	D8	ABS
DFA5	D7	INT
E22A	DE	SQR
E234	D0	↑
E2A6	E1	EXP
E34B	DF	RND
E387	E2	COS
E38E	E3	SIN
E3D7	E4	TAN
E43B	E5	ATN
(E76B)	C7	AUTO
E7BA	B6	CLOAD
E7DB	B7	CSAVE
E80D	BF	CALL
E95B	9E	HIMEM
E974	9F	GRAB
E994	A0	RELEASE
E9CD	F3	POINT
F02D	AA	CURSET
F064	AB	CURMOV
F079	AC	DRAW
F093	AE	PATTERN
F0A5	B0	CHAR
F17F	B1	PAPER
F18B	B2	INK
F1E5	AF	FILL
F2E5	AD	CIRCLE
F923	A1	TEXT
FA85	A6	PING
FA9B	A3	SHOOT
FAB1	A4	EXPLODE
FAC7	A5	ZAP
FB26	A7	SOUND
FBB6	A9	PLAY
FBE3	A2	HIRES
FBFE	A8	MUSIC

LISTE DES ADRESSES DES FONCTIONS PAR ADRESSES. V.1.1.

CODE	FONCTION	ADRESSES
D9	USR	0021
DD	&	02FB
81	EDIT	C692
C1	NEW	C6EE
BD	CLEAR	C70D
BC	LIST	C748
8E	LLIST	C7FD
8F	LPRINT	C809
8D	FOR	C855
C3	TO	(C880)
CB	STEP	(C8A8)
9A	RESTORE	C952
B3	STOP	C971
80	END	C973
BB	CONT	C9A0
98	RUN	C9BD
9B	GOSUB	C9C8
97	GOTO	C9E5
86	POP	CA12
9C	RETURN	CA12
91	DATA	CA3C
99	IF	CA70
C9	THEN	(CA7C)
9D	REM	CA99
C8	ELSE	(CAA9)
B4	ON	CAC2
96	LET	CB1C
BA	PRINT	CBAB
C6	à	(CBC7)
C2	TAB(	(CC2E)
C5	SPC(	(CC2E)
94	CLS	CCCE
84	TRON	CD16
85	TROFF	CD19
BE	GET	CD46
92	INPUT	CD55
95	READ	CD89
90	NEXT	CE98
CA	NOT	D03C
D2	OR	D0E3
D1	AND	D0E6
D3	>	D113
D4	=	D113
D5	<	D113
93	DIM	D17E
DA	FRE	D47E
DB	POS	D4A6
B8	DEF	D4BA
C4	FN	(D50D)
EA	STR$	D593
ED	CHR$	D816
F4	LEFT$	D82A
F5	RIGHT$	D856
F6	MID$	D861
E9	LEN	D8A6
EC	ASC	D8B5
EB	VAL	D8D7
E6	PEEK	D938
B9	POKE	D94F
B5	WAIT	D958
8A	DOKE	D967
E7	DEEK	D983
DC	HEX$	D9B5
89	LORES	D9DE
F2	SCRN	DA3F
87	PLOT	DA51
8B	REPEAT	DA85
88	PULL	DAA1
8C	UNTIL	DAA1
F1	KEY$	DADA
CD	-	DB0E
CC	+	DB25
E0	LN	DCAF
CE	*	DCF0
E8	LOG	DDD4
CF	/	DDE7
EE	PI	DE77
F0	FALSE	DF0B
EF	TRUE	DF0F
D6	SGN	DF21
D8	ABS	DF49
D7	INT	DFBD
DE	SQR	E22E
D0	↑	E238
E1	EXP	E2AA
DF	RND	E34F
E2	COS	E38B
E3	SIN	E392
E4	TAN	E3DB
E5	ATN	E43F
C7	AUTO	(E7FE)
B6	CLOAD	E85B
B7	CSAVE	E909
BF	CALL	E946
82	STORE	E987
83	RECALL	E9D1
9E	HIMEM	EBCE
9F	GRAB	EBE7
A0	RELEASE	EC0C
A1	TEXT	EC21
A2	HIRES	EC33
F3	POINT	EC45
C0	!	ED13
AA	CURSET	F0C8
AB	CURMOV	F0FD
AC	DRAW	F110
AE	PATTERN	F11D
B0	CHAR	F12D
B1	PAPER	F204
B2	INK	F210
AF	FILL	F268
AD	CIRCLE	F37F
A6	PING	FA9F
A3	SHOOT	FAB5
A4	EXPLODE	FACB
A5	ZAP	FAE1
A7	SOUND	FB40
A9	PLAY	FBD0
A8	MUSIC	FC18

LISTE DES ADRESSES DES FONCTIONS PAR ORDRE ALPHANUMERIQUE. V.A.O.

FONCTION	CODE	ADRESSES
!	C0	CC89
&	DD	02FB
*	CE	DCBA
+	CC	DA9A
-	CD	DA83
/	CF	DDE3
<	D5	D087
=	D4	D087
>	D3	D087
↑	D0	E234
à	C6	----
ABS	D8	DF31
AND	D1	D05A
ASC	EC	D7FA
ATN	E5	E43B
AUTO	C7	(E76B)
CALL	BF	E80D
CHAR	B0	F0A5
CHR$	ED	D75B
CIRCLE	AD	F2E5
CLEAR	BD	C73A
CLOAD	B6	E7BA
CLS	94	CC0A
CONT	BB	C970
COS	E2	E387
CSAVE	B7	E7DB
CURMOV	AB	F064
CURSET	AA	F02D
DATA	91	CA09
DEEK	E7	D8C8
DEF	B8	D401
DIM	93	D0F2
DOKE	8A	D8AC
DRAW	AC	F079
EDIT	81	C6A5
ELSE	C8	(CA6D)
END	80	C941
EXP	E1	E2A6
EXPLODE	A4	FAB1
FALSE	F0	DEFC
FILL	AF	F1E5
FN	C4	(D454)
FOR	8D	C841
FRE	DA	D3D6
GET	BE	CCBA
GO	F7	----
GOSUB	9B	C996
GOTO	97	C9B3
GRAB	9F	E974
HEX$	DC	D917
HIMEM	9E	E95B
HIRES	A2	FBE3
IF	99	CA3E
INK	B2	F18B
INPUT	92	CCC9
INT	D7	DFA5
INVERSE	82	CFE4
KEY$	F1	DA4F
LEFT$	F4	D76F
LEN	E9	D7EB

FONCTION	CODE	ADRESSES
LET	96	CAD2
LIST	BC	C773
LLIST	8E	C824
LN	E0	DC79
LOG	E8	DDD0
LORES	89	D937
LPRINT	8F	C832
MID$	F6	D7A6
MUSIC	A8	FBFE
NEW	C1	C71B
NEXT	90	CE0C
NORMAL	83	CFE4
NOT	CA	CFB0
ON	B4	CA78
OR	D2	D057
PAPER	B1	F17F
PATTERN	AE	F093
PEEK	E6	D87D
PI	EE	D8EE
PING	A6	FA85
PLAY	A9	FBB6
PLOT	87	D9C6
POINT	F3	E9CD
POKE	B9	D894
POP	86	C9E0
POS	DB	D3FA
PRINT	BA	CB61
PULL	88	DA16
READ	95	CCFD
RELEASE	A0	E994
REM	9D	CA61
REPEAT	8B	D9FA
RESTORE	9A	C91F
RETURN	9C	C9E0
RIGHT$	F5	D79B
RND	DF	E34B
RUN	98	C990
SCRN	F2	D9B4
SGN	D6	DF12
SHOOT	A3	FA9B
SIN	E3	E38E
SOUND	A7	FB26
SPC(	C5	(CBCA)
SQR	DE	E22A
STEP	CB	(C894)
STOP	B3	C93F
STR$	EA	D4D8
TAB(	C2	(CBCA)
TAN	E4	E3D7
TEXT	A1	F923
THEN	C9	(CA48)
TO	C3	(C86C)
TROFF	85	CC8F
TRON	84	CC8C
TRUE	EF	DF00
UNTIL	8C	DA16
USR	D9	0021
VAL	EB	D81C
WAIT	B5	D89D
ZAP	A5	FAC7

LISTE DES ADRESSES DES FONCTIONS PAR ORDRE ALPHANUMERIQUE. V.A.A.

CODE	FONCTION	ADRESSES
C0	!	ED13
DD	&	02FB
CE	*	DCF0
CC	+	DB25
CD	-	DB0E
CF	/	DDE7
D5	<	D113
D4	=	D113
D3	>	D113
D0	↑	E238
C6	à	(CBC7)
D8	ABS	DF49
D1	AND	D0E6
EC	ASC	D8B5
E5	ATN	E43F
C7	AUTO	(E7FE)
BF	CALL	E946
B0	CHAR	F12D
ED	CHR$	D816
AD	CIRCLE	F37F
BD	CLEAR	C70D
B6	CLOAD	E85B
94	CLS	CCCE
BB	CONT	C9A0
E2	COS	E38B
B7	CSAVE	E909
AB	CURMOV	F0FD
AA	CURSET	F0C8
91	DATA	CA3C
E7	DEEK	D983
B8	DEF	D4BA
93	DIM	D17E
8A	DOKE	D967
AC	DRAW	F110
81	EDIT	C692
C8	ELSE	(CAA9)
80	END	C973
E1	EXP	E2AA
A4	EXPLODE	FACB
F0	FALSE	DF0B
AF	FILL	F268
C4	FN	(D50D)
8D	FOR	C855
DA	FRE	D47E
BE	GET	CD46
9B	GOSUB	C9C8
97	GOTO	C9E5
9F	GRAB	EBE7
DC	HEX$	D9B5
9E	HIMEM	EBCE
A2	HIRES	EC33
99	IF	CA70
B2	INK	F210
92	INPUT	CD55
D7	INT	DFBD
F1	KEY$	DADA
F4	LEFT$	D82A
E9	LEN	D8A6
96	LET	CB1C
BC	LIST	C748

CODE	FONCTION	ADRESSES
8E	LLIST	C7FD
E0	LN	DCAF
E8	LOG	DDD4
89	LORES	D9DE
8F	LPRINT	C809
F6	MID$	D861
A8	MUSIC	FC18
C1	NEW	C6EE
90	NEXT	CE98
CA	NOT	D03C
B4	ON	CAC2
D2	OR	D0E3
B1	PAPER	F204
AE	PATTERN	F11D
E6	PEEK	D938
EE	PI	DE77
A6	PING	FA9F
A9	PLAY	FBD0
87	PLOT	DA51
F3	POINT	EC45
B9	POKE	D94F
86	POP	CA12
DB	POS	D4A6
BA	PRINT	CBAB
88	PULL	DAA1
95	READ	CD89
83	RECALL	E9D1
A0	RELEASE	EC0C
9D	REM	CA99
8B	REPEAT	DA85
9A	RESTORE	C952
9C	RETURN	CA12
F5	RIGHT$	D856
DF	RND	E34F
98	RUN	C9BD
F2	SCRN	DA3F
D6	SGN	DF21
A3	SHOOT	FAB5
E3	SIN	E392
A7	SOUND	FB40
C5	SPC(	(CC2E)
DE	SQR	E22E
CB	STEP	(C8A8)
B3	STOP	C971
82	STORE	E987
EA	STR$	D593
C2	TAB(	(CC2E)
E4	TAN	E3DB
A1	TEXT	EC21
C9	THEN	(CA7C)
C3	TO	(C880)
85	TROFF	CD19
84	TRON	CD16
EF	TRUE	DF0F
8C	UNTIL	DAA1
D9	USR	0021
EB	VAL	D8D7
B5	WAIT	D958
A5	ZAP	FAE1



BONJOUR LES MICRODISQUES

par Fabrice BROCHE

Ça y est, les drives tant attendus sont enfin arrivés. Malheureusement, et comme à leur habitude, les ingénieurs d'ORIC sont très discrets du côté du fonctionnement interne. Je vais donc vous faire part de mes découvertes (mai 1984).

Le 6502 considère tout périphérique comme un ensemble de cases mémoires : ainsi le 6522 s'implante dans la zone # 30Ø à # 3ØF. Bien entendu c'est la même chose pour le contrôleur de disquettes. Celui-ci prend place aux octets # 31Ø à # 31B comme suit :

# 31Ø	registre de commande
# 311	registre de piste
# 312	registre de secteur
# 313	registre d'entrée/sortie
# 314	registre du contrôleur
# 315-317	et # 319-#31B ?? peut-être pour l'utilisation de plusieurs drives?
#318	registre du contrôleur

Nous reviendrons plus loin sur la signification de ces registres.

D'autre part, l'ORIC a 64 K de RAM et 16 K de ROM ce qui fait 80 K... trop pour être adressable directement par le 65Ø2 qui ne peut adresser que 64 K de mémoire.

D'où une astuce : mettre le 16 K de RAM **en overlay** : c'est-à-dire masqués par le BASIC (#CØØØ à # FFFF). On peut accéder alors à la RAM, aux mêmes adresses si on n'utilise pas le BASIC. On dit qu'on a 2 pages de 16 K : en l'occurence une page de ROM et une page de RAM.

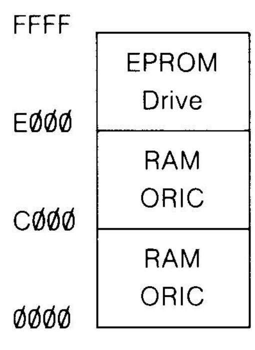
Pour déconnecter la ROM et accéder à la RAM, il faut agir correctement sur les signaux MAP et ROM DIS du BUS, ce qu'à l'heure actuelle seuls les concepteurs de systèmes de disquettes savent faire.

Donc, le DOS se charge dans les 16 K de RAM masqués par le BASIC.

D'autre part, ceux qui ont vu l'intérieur d'un microdrive (attention! si cela vous tente... vous perdez la garantie) auront découvert une EPROM de 8 K. Cette mémoire permanente contient le programme d'amorçage : celui qui affiche **"Insert system disc..."**.

Essayons de voir ce qui se passe entre le moment où on appuie sur le "RESET" du Microdrive et où l'ordinateur nous donne la main.

Tout laisse à penser qu'à cet instant l'ORIC passe non seulement sur la RAM de # CØØØ à # DFFF, mais sur l'EPROM de # EØØØ à # FFFF autrement dit, voici la carte mémoire à cet instant :



L'ORIC exécute donc le programme de l'EPROM. Celui-ci transfère en RAM deux routines :

- en # 48Ø-#4FF celle qui permettra tous les passages RAM/ROM
- en # BFEØ-#BFFA la routine d'amorçage.

Le programme contenu en EPROM effectue quelques tests. En particulier le contrôle du circuit contenant une résistance variable situé dans le lecteur. Cette résistance règle probablement la tension sur la broche MAP. Lorsqu'elle est mal réglée, le contrôleur n'arrive pas à passer sur la RAM. Un message s'affiche **"RV1 adjustment**

required". N'essayez pas de faire apparaître ce message en déréglant la résistance !

Ensuite, il affiche **"Insert system disc".** Il effectue alors une routine probablement très proche du !LOAD. Il recherche le programme dénommé "SYSTEM DOS". Lorsque celui-ci est chargé, le programme en EPROM exécute une routine d'initialisation assez semblable à un RESET à froid mais interne à l'EPROM puisqu'elle ne vide pas la mémoire, etc...

Le programme sur EPROM effectue un saut en # BFEØ non sans avoir placé en # 35 (c'est-à-dire dans le tampon du clavier) tel quel !BOOT UP.

Cette routine passe alors sur la RAM et élimine ainsi l'EPROM puis exécute un saut en # C14B-C où se trouve l'adresse d'exécution du système d'exploitation de disquettes (ou DOS).

Exécuter le système DOS, cela consiste à décaler le programme chargé en # 74ØØ- # AØ34 vers # D4ØØ- # FFFF non sans avoir effectué un test pour savoir si le DOS est bien chargé.

Dans le cas contraire, le programme va "boucler".

A ce moment la carte mémoire se présente ainsi :

Adresse	Zone
FFFF	
	DOS
D4ØØ	
	RAM
BFF5	
	Amorce
BFEØ	
	RAM
AØ35	
	System DOS
74ØØ	
	RAM
4FF	
	Amorce
48Ø	
	RAM
ØØ	

(échelle non respectée)

Après exécution du "SYSTEM DOS" la routine effectue un saut à l'interpréteur pour que celui-ci déclenche une !BOOT UP, rendant la main après exécution.

Si vous êtes incrédules, utilisez votre moniteur, entrez le programme qui suit et sauvez le par !SAVE "TEST-COM", A # 4ØØ, E # 437, T # 415.

```
0400  85 02     STA #02         SOUS PROGRAMME DECALAGE.
0402  84 03     STY #03
0404  A0 00     LDY %#00
0406  B1 00     LDA (#00),Y
0408  91 02     STA (#02),Y
040A  88        DEY
040B  D0 F9     BNE #0406
040D  E6 01     INC #01
040F  E6 03     INC #03
0411  CA        DEX
0412  10 F2     BPL #0406
0414  60        RTS
0415  A9 00     LDA %#00
0417  A0 00     LDY %#00
0419  85 00     STA #00
041B  84 00     STY #00         Décaler à partir de 0000.
041D  A0 10     LDY %#10        vers #1000.
041F  A2 05     LDX %#05        6 pages.
0421  20 00 04  JSR #0400
0424  A9 00     LDA %#00
0426  A0 C0     LDY %#C0        Décaler à partir de #C000.
0428  85 00     STA #00
042A  84 01     STY #01
042C  A0 20     LDY %#20        Vers #2000.
042E  A2 3F     LDX %#3F        64 pages.
0430  20 00 04  JSR #0400
0433  BA        TSX
0434  8E 00 60  STX #6000       et sauver le pointeur de pile.
0437  60        RTS
```

Le but de ce programme est simple : décaler les pages Ø à 4, de (#Ø à # 4FF) en (#1ØØØ à 14FF), décaler également l'emplacement (#CØØØ à # FFFF) en (#2ØØØ à # 5FFF), et de sauver le pointeur de pile en # 6ØØØ.

Vous avez sûrement une disquette disponible, dépourvue de "SYSTEM DOS", sauvez donc notre programme sous ce nom en faisant :
!COPY "TEST . COM" TO "SYSTEM.DOS" (,C)

Éteignez l'ORIC. C'est le seul moyen de vider totalement la mémoire à coup sûr. Remettez sous tension. Tentez une initialisation du DOS : la tentative échoue... et c'est normal puisque le vrai SYSTEM.DOS n'est plus là, toutefois cette manœuvre a chargé nos parties mémoires, nous allons pouvoir observer tout cela bientôt.

EN SAVOIR PLUS SUR LE D.O.S.

Insérez un vrai "SYSTEM.DOS" dans le lecteur et initialisez. Chargez un **MONITEUR** pour contrôler nos commentaires.

Observation de la page Ø (de #1ØØØ à #1ØFF).

Outre quelques vecteurs et pointeurs initialisés nous voyons en # 36 : !BOOT UP (le point d'exclamation a disparu si votre disquette est dépourvue de BOOT UP). C'est bien ce qui avait été annoncé.

Voilà une excellente occasion de savoir le minimum que nécessite l'ORIC pour fonctionner, mais nous sortons du sujet qui nous occupe ici.

Intéressons nous à la pile. Elle a été remplie jusqu'en # 11CA. Le pointeur de pile indique # FC. En # 11FD- #11FE nous trouvons donc l'adresse 1 d'où a été appelé le SYSTEM.DOS, s'il a été appelé par sous programme cette adresse est # BFEC : la routine d'amorce. Nous y reviendrons.

Notons qu'on reconnaît sur la pile la trace des interruptions, soit pour l'ATMOS :
Ø 6F EE ØE DD 4Ø 2F EE 31

Rien à signaler de particulier sauf une certaine similitude avec la trace laissée par un !LOAD normal : l'EPROM et le DOS sont sûrement très proches !

Sautons en # 14ØØ : nous y voyons notre routine, c'est normal, puis en #148Ø : la routine y a déjà été implantée.

Voyons du côté de la RAM **en overlay,** translatée ici en # 2ØØØ.

#2ØØØ	Ø nous sommes sur le lecteur/enregistreur Ø
#2ØØ1	piste } correspondant au répertoire où a été trouvé le SYSTEM.DOS
#2ØØ2	secteur }
#2ØØ3	} contient l'adresse où à été chargé le répertoire : # 2Ø23
#2ØØ4	}
#2Ø23	le répertoire, on peut y repérer "SYSTEM.DOS"
#212C	"BOOTUP.COM" : c'est bien là que la routine du DOS l'aurait mis
#214B (équivalent de #C14B)	#415 adresse d'exécution de notre "SYSTEM.DOS"

Ensuite, plus rien, la RAM est vide!!. Voyons de plus près la routine d'amorce :

```
BFEØ  78        SEI             SUPPRIMER IRQ.
BFE1  A9 84     LDA %#84        Passer sur la RAM et éliminer l'EPROM.
BFE3  8D 8Ø Ø4  STA #Ø48Ø       Sauver pour prochains échanges.
BFE6  8D 14 Ø3  STA #Ø314       Et effectuer opération.
BFE9  2Ø F8 BF  JSR #BFF8       exécuter SYSTEM.DOS.
BFEC  A2 34     LDX %#34        préparer pointeur pour programme.
BFEE  AØ ØØ     LDY %#ØØ        exécuter tampon clavier

BFFØ  58        CLI             Restaurer IRQ.
BFF1  2Ø 5A D4  JSR #D45A       et exécuter !BOOTUP s'il existe. Rien sinon.

BFF4  CD C4 BD C4               Adresses routines ROM V1.Ø et V1.1

BFF8  6C 4B C1  JMP (#C14B)
```

Voilà, c'est à peu près tout sur la genèse du DOS. Une étape qui permettrait de tout savoir serait de lister l'EPROM : c'est sûrement possible. A première vue seuls 4 K sur les 8 disponibles sont utilisés D'autres versions du DOS vont sans aucun doute venir. Élaborer un bon DOS n'a pas l'air simple...

Dans un prochain article nous verrons comment exploiter toutes ces possibilités. En attendant pour permettre aux plus curieux d'entre vous d'explorer seuls, voici quelques indications :

pour passer sur la RAM :

PHP
LDA % #ØØ
JSR # 4E6
PLP

sur la ROM :

PHP
LDA % # Ø2
JSR # 4E6
PLP

pour exécuter une routine, tout en étant sur la RAM :

DOKE # 485, adresse : CALL #49Ø

Voici les variables système, peu nombreuses, du DOS :

# CØØØ	n° du lecteur courant
# CØØ1	n° de piste
# CØØ2	n° du secteur
# CØØ3-4	adresse de chargement du secteur (normal : # Ø23)
# CØØ7	Ø : ROM V1.Ø 1 : ROM V1.1
# CØØC	lecteur/enregistreur par défaut (instruction DRV)
# CØ13-4-5-6	pour chaque lecteur, nombre de pistes
# CØ17-8-9-A	pour chaque lecteur, ØØ : simple face 8Ø : double face

Pour lire un secteur :

Positionner correctement # CØØØ-1-2-3-4 puis faire JSR # D424.

Pour écrire dans un secteur :

JSR # D421.

Notez encore que le point d'exclamation est exécuté en # 4C4.

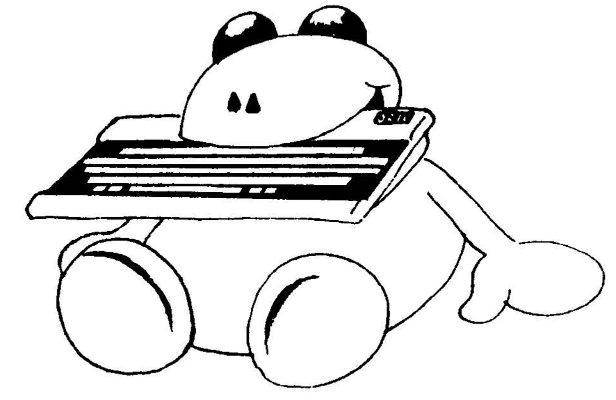
Le listing commenté de la routine de dialogue RAM/ROM (# 484- # 4FA) est donné ci-après :

```
#480   état courant du contrôleur.
#481   travail.
#482   travail sauver A.
#483   travail sauver P.
#4FD   autoriser l'affichage des erreurs.
#4FE   détection des erreurs d'e/s directement affichée par FDERR.
#4FF   n° de l'erreur.
```

```
0484  4C 66 D4 JMP #D466     Vecteur exécution.
0487  4C E6 04 JMP #04E6     Vecteur passage RAM/ROM.
048A  4C D6 04 JMP #04D6     Vecteur IRQ.
048D  4C DE 04 JMP #04DE     Vecteur NMI.
0490  08          PHP        Sauver IRQ éventuellement.
0491  78          SEI        Interdire IRQ.
0492  8D 82 04 STA #0482     Sauver A.
```

```
0495  68        PLA         Récupérer registre d'état.
0496  8D 83 04  STA #0483   pour le sauver.
0499  AD 80 04  LDA #0480   Sauver de même
049C  48        PHA         l'état actuel.
049D  AD 81 04  LDA #0481   Récupérer l'ordre 0. Passer sur la RAM - 2 ROM.
04A0  20 E6 04  JSR #04E6
04A3  AD 83 04  LDA #0483
04A6  48        PHA         Empiler ancien registre d'état.
04A7  AD 82 04  LDA #0482   Récupérer A.
04AA  28        PLP         et le registre d'état.
04AB  20 84 04  JSR #0484   Exécuter routine demandée.
04AE  08        PHP         Resauver registre d'état.
04AF  78        SEI         Supprimer à nouveau IRQ.
04B0  8D 82 04  STA #0482   Resauver A.
04B3  68        PLA
04B4  8D 83 04  STA #0483   Resauver registre d'état.
04B7  68        PLA         Récupérer ancienne position.
04B8  20 E6 04  JSR #04E6   Et restaurer l'état initial.
04BB  AD 83 04  LDA #0483
04BE  48        PHA
04BF  AD 82 04  LDA #0482
04C2  28        PLP         et récupérer registre d'état et A.
04C3  60        RTS
04C4  A9 00     LDA %#00         GESTION DU '!'
04C6  8D 81 04  STA #0481   Indiquer passage sur la RAM.
04C9  A9 66     LDA %#66
04CB  8D 85 04  STA #0485   Préparer vecteur
04CE  A9 D4     LDA %#D4
04D0  8D 86 04  STA #0486   pour saut en #D466.
04D3  4C 90 04  JMP #0490   e  exécuter saut en #D466 en RAM
04D6  08        PHP         puis revenir au BASIC.
04D7  BA        TSX
04D8  FE 02 01  INC #0102,X Sauter le PMP (#4AE)...Bizarre!
04DB  4C 44 02  JMP #0244   Exécuter IRQ de la ROM.
04DE  08        PHP
04DF  BA        TSX
04E0  FE 02 01  INC #0102,X Sauter le PMP (#4AE)
04E3  4C 47 02  JMP #0247   Exécuter NMI de la ROM.
04E6  78        SEI         Supprimer IRQ.
04E7  29 02     AND %#02    Isoler le bit RAM/ROM.
04E9  8D 81 04  STA #0481   et le sauver pour travail.
04EC  AD 80 04  LDA #0480   Recopier ancien état.
04EF  29 FD     AND %#FD    Isoler le bit RAM/ROM.
04F1  0D 81 04  ORA #0481   Et y placer la valeur désirée.
04F4  8D 14 03  STA #0314   Exécuter ordre.
04F7  8D 80 04  STA #0480   Et sauver nouvel état.
04FA  60        RTS
```

De plus, je vous livre ce que je sais des registres # 310-# 318. Ces indications sont le fruit de mes découvertes puisque je ne possède pas le schéma du contrôleur. Toutefois tout a été vérifié avec autant de minutie que possible.



DOS : REGISTRES D'ENTRÉES/SORTIES

Notation	7	6	5	4	3	2	1	Ø

■ **314** Registre du contrôleur de disquette.

Ce registre est utilisé à la fois en entrée et en sortie. Il n'est probablement pas "latché" (verrouillé) ce qui expliquerait que sa lecture donne toujours # FF ainsi que l'obligation de sauver sa valeur en # 48Ø.

Une autre explication serait que sa configuration normale soit en entrée et que la sortie n'apparaisse que le temps où la ligne R/W est à l'état W.

En entrée :

7 IRQ (active et autorisée à Ø)
Cette IRQ est générée sur un signal "Ready" (se reporter au bit Ø de l'octet 31Ø).

En sortie :

Ø Demande d'interruption et traitement de l'interruption IRQ par mise à Ø. (1er point sous réserves).

1 ROM DIS sélection de la ROM : Ø ROM inhibée
1 ROM sélectionnée

2
3 apparemment non utilisé

4 "Side select" Ø simple face ou face A sur une disquette double face.
1 face B d'une disquette double face.

5	n° du drive	Ø / Ø : drive 1	1 / Ø : drive 2	Ø / 1 : drive 3	1 / 1 : drive 4
6	à activer				

7 EPROM select Ø EPROM sélectionné
(attention! ROM DIS est "prioritaire") donne accès à l'EPROM du contrôleur de # EØØØ à # FFFF
1 EPROM inhibée

■ **# 318**

En entrée :

7 "Data Ready" Ø on peut lire en # 313, ou écrire la donnée lors des procédures RWTS
[RWTS = Read Write Track Sector
Lecture Écriture Piste Secteur]

■ **# 31Ø** Registre de commande du lecteur/enregistreur.
Ce registre est utilisé en entrée et en sortie. Cette fois, la fonction entrée est "latchée".

En entrée :

6 "Write protected" 1 disquette protégée en écriture
Ø disquette non protégée

5 Positionnement tête (sous réserve)

4 Impossibilité lecture/écriture (non formattée ?)

3 Erreur dans les procédures R/W (?)

2 La tête se trouve sur la piste Ø
1

Ø Opération terminée : 256 octets d'un secteur lus, tête positionnée, etc...
Lorsqu'il passe à Ø, une IRQ peut être générée, permettant normalement de sortir des routines RWTS.

En sortie :

L'utilisation du registre de commande est assez complexe ; il est difficile d'affecter une signification précise à chaque bit.

5 Bit de direction à Ø : **lecture,** ou déplacement de la tête vers le centre
à 1 : **écriture** ou déplacement de la tête vers l'extérieur
ce bit sera noté S.

4 Autorisation de modifier le compteur de piste lors des déplacements de la tête (?).
Ce bit sera noté C.
Un bit est noté X s'il semble ne pas avoir d'effet.

Déplacement de la tête

Ø	1	S	C	1	X	X	X

Par exemple # 58 déplace la tête vers le centre en incrémentant # 311.

Placer la tête piste Ø

Ø	Ø	Ø	Ø	1	Ø	Ø	Ø

Lecture/écriture

1	Ø	S	Ø	X	X	Ø	Ø

peut lire ou écrire en # 313.

Formatter

1	1	1	1	Ø	Ø	Ø	Ø

Pour # 313 : comme ci-dessus.

N.B. : La capacité non formattée d'une piste est d'environ 6 K.

Demande d'information

1	1	S	Ø	Ø	Ø	Ø	Ø

Envoie 6 octets VIA # 313
312 contient le n° de la piste courante.

Parmi les 6 octets donnés

1 n° de piste

2 Ø simple face 1 face B

3 n° de secteur

4 longueur du secteur
Ø : 128 octets
1 : 256 octets
2 : 512 octets
3 : 1 Ko

5-6 probablement contrôle de la validité des informations du secteur (une fonction logique liant tous les octets du secteur ?).

Ces 6 octets forment l'en-tête d'un secteur.

Pour positionner directement la tête sur une piste

Ø	Ø	Ø	1	1	1	Ø	Ø	Ø

le n° de piste étant en # 313.

Fabrice BROCHE

Oricment vôtre

COURRIER DES LECTEURS

M. MORIN Marcel de St Etienne
propose un RENUMÉROTEUR simple en langage machine

A la suite de l'article du dernier MICR'ORIC ; concernant le renumérotage de lignes basic. J'ai fait mes premiers pas en langage machine et je vous propose ce petit programme simple de renumérotage de 10 en 10.

Avantage

On peut l'utiliser directement et ce à tout moment dans un programme BASIC en faisant (!) + Return.

On peut le changer à tout moment par CLOAD "." si préalablement on l'a sauvegardé par CSAVE " " ,A #4ØØ ,E #44Ø temps de lecture et chargement (2 secondes).

Si on fait des NEW ou RESET rien n'est perdu de ce programme.

Inconvénient

Je n'ai pas encore regardé comment renuméroter les GOSUB, GOTO et THEN. Il faut donc noter le n° de ligne sur laquelle ils pointent et la multiplier par 10 (n° ligne + 10). Faire la correction et ensuite (!) + Return.

```
10 REM RENUMEROTATION SIMPLE
20 READ A$
30 A$="#"+A$:A=VAL(A$)
35 IFA=#FFTHEN END
40 POKE #400+B,A
50 B=B+1
60 GOTO 10
70 DOKE #2F5,#400
80 DATA A9,00,8D,75,04,8D,76,04
90 DATA A5,9A,A6,9B,8D,77,04,8E
100 DATA 78,04,A0,00,20,53,04,8D
110 DATA 79,04,C8,20,53,04,48,18
120 DATA 6D,79,04,B0,02,F0,54,68
130 DATA 8D,7A,04,A9,0A,18,6D,75
140 DATA 04,90,01,EE,76,04,8D,75
150 DATA 04,C8,20,63,04,AD,76,04
160 DATA C8,20,63,04,AD,79,04,AE
170 DATA 7A,04,8D,77,04,8E,78,04
180 DATA 4C,12,04,AD,77,04,8D,60
190 DATA 04,AD,78,04,8D,61,04,B9
200 DATA 3E,05,60,48,AD,77,04,8D
210 DATA 72,04,AD,78,04,8D,73,04
220 DATA 68,99,38,05,60,00,00,00
230 DATA 00,00,00,68,60,FF
```

EXPLICATIONS

ADRESSES

```
#400-40F   Initialisation et chargement des registres.
#412-41B   Chargement des nouvelles adresses.
#41E-41B   Test si la nouvelle adresse=0 alors fin.
#428-436   Addition des n° de lignes.
#439-441   Ecriture de la nouvelle valeur de lignes.
#444-450   Adresse nouvelle = Ancienne.
#453-462   Routine de chargement d'une adresse.
#47B-47C   Sortie du programme.
#475-476   Résultat d'addition.
#477-478   Adresse de début de ligne (ordre n).
#479-480   Adresse de début de ligne (ordre n+1).
```

Le chargement se fait à tout moment par CLOAD"".

Faire DOKE #2F5,#400 puis '!' et <Return>

Le pas entre les lignes est de 10 mais il est possible
de le faire varier de 0 à 255 (0<M<255)
en faisant POKE 1068,M puis '!' et <Return>.

```
#9A-#9B  Adresse du 1er octet BASIC.

Dans les octets #460-#461-#472-#473
on peut mettre ce qu'on veut puisque leur contenu
sera modifié à l'exécution du programme.
```

Mnémoniques utilisées

```
0400 A900   LDA %#00
0402 8D7504 STA #0475
0405 8D7604 STA #0476
0408 A59A   LDA #9A
040A A69B   LDX #9B
040C 8D7704 STA #0477
040F 8E7804 STX #0478
0412 A000   LDY %#00
0414 205304 JSR #0453
0417 8D7904 STA #0479
041A C8     INY
041B 205304 JSR #0453
041E 48     PHA
041F 18     CLC
0420 6D7904 ADC #0479
0423 B002   BCS #0427
0425 F054   BEQ #047B
0427 68     PLA
0428 8D7A04 STA #047A
042B A90A   LDA %#0A
042D 18     CLC
042E 6D7504 ADC #0475
0431 9001   BCC #0434
0433 EE7604 INC #0476
0436 8D7504 STA #0475
0439 C8     INY
043A 206304 JSR #0463
043D AD7604 LDA #0476
0440 C8     INY
0441 206304 JSR #0463
0444 AD7904 LDA #0479
0447 AE7A04 LDX #047A
044A 8D7704 STA #0477
044D 8E7804 STX #0478
0450 4C1204 JMP #0412
0453 AD7704 LDA #0477
0456 8D6004 STA #0460
0459 AD7804 LDA #0478
045C 8D6104 STA #0461
045F B93E05 LDA #053E,Y
0462 60     RTS
0463 48     PHA
0464 AD7704 LDA #0477
0467 8D7204 STA #0472
046A AD7804 LDA #0478
046D 8D7304 STA #0473
0470 68     PLA
0471 993805 STA #0538,Y
0474 60     RTS
0475 00     BRK
0476 00     BRK
0477 00     BRK
0478 00     BRK
0479 00     BRK
047A 00     BRK
047B 68     PLA
047C 60     RTS
047D 68     PLA
047E 60     RTS
047F 5586   EOR #86,X
0481 02     ???
0482 00     BRK
0483 73     ???
0484 4C65C7 JMP #C765
0487 4CE604 JMP #04E6
048A 4CD604 JMP #04D6
048D 4CDE04 JMP #04DE
0490 08     PHP
```

M. MORIN après avoir lu "CHIRURGIE EN RAM" (MICR'ORIC n° 2) ayant compris comment était représenté un programme BASIC en RAM, puis dans MICR'ORIC n° 3 ayant vu les propositions de programme de RENUMÉROTATION en langage BASIC a eu envie de créer le programme correspondant en langage machine.

Un tel programme n'étant pas affecté par un NEW est plus commode d'emploi.

Le savoir faire s'acquiert peu à peu. Voici les sources utilisées :

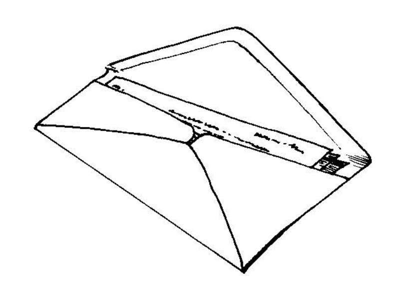
- MICRO & ROBOTS (articles sur le 6502) n°1 à 4.
- ASSEMBLEUR/DÉSASSEMBLEUR de Ph. G. dans MICRO SYSTÈME
- LA PROGRAMMATION du 6502 Édition SYBEX
- VISA pour l'ORIC (pour les adresses)

M. MORIN s'est mis à la microinformatique depuis 6 mois. Après avoir programmé sur un PR 100 (CASIO) il a poursuivi sur ORIC.

Vous êtes très nombreux à vous former en quasi autodidactes. Les témoignages de vos efforts sont de nature à encourager les isolés. Écrivez-nous comme M. MORIN, nous publierons des extraits de vos travaux.

M. LE RÉVÉREND Rémi
59122 HONDSCHOOTE

nous écrit :



"Je vous écrit au sujet de l'article paru dans l'ordinateur Individuel n° 58 (avril 84) dans la rubrique Coq-Oric-Oh. Cet article traite des routines Basic.

En effet, après avoir trouvé les codes des instructions grâce à "Chirurgie en RAM" (Microric n°2) j'ai cherché les adresses des routines du Basic qui se calculent par la formule :

HEX$ (DEEK (# CØØ + (Code-128) * 2) + 1)

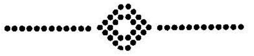
Tout fonctionne bien dans la plupart des cas mais, par exemple pour les commandes de code 170 à 176, c'est-à-dire tout le graphisme en haute résolution, on retombe toujours sur le même code : # E87D. Il se produit le même phénomène pour les commandes SOUND, MUSIC, PLAY, PAPER et INK où l'adresse est # E889. A quoi est-ce dû et existe-t-il un moyen de trouver les codes corrects ?

D'autre part, la procédure DOKE # 2F5, adresse qui assigne une fonction (désignée par son adresse au point d'exclamation, fonctionne pour les ordres qui ne nécessitent aucun paramètre, mais pour les autres (PLOT, LORES, GOTO, GOSUB, etc.), quelle syntaxe adopter ?

Troisièmement, on obtient, quand on n'a rien assigné, certaines adresses pour des fonctions non utilisables (à ma connaissance) :

130	INVERSE	# CFE4
131	NORMAL	# CFE4
192	!	# CC89
198	ծ	# D3D7
221	&	# DA50
246	GO	# 5246

Quand on tente de les utiliser, les deux premières donnent ?SYNTAX ERROR et les autres bloquent l'ordinateur. Pourriez-vous éclaircir la signification de ces adresses ainsi que leur utilisateur, si elle est possible ?



Enfin, pour plusieurs instructions, on obtient des codes fantaisistes qui n'appartiennent pas à la ROM. Voici la liste de ces instructions :

STR$	# 5ØE3	TAN	# 79DB	VAL	# D05A
SPC	# ØØ22	PEEK	# B97C	ASC	# 5647
THEN	# Ø2FC	DEEK	# 7BDD	CHR$	# 7DD1
SIN	# 997A	LEN	# 338Ø	TRUE	# AF5B
FALSE	# 64DØ	POINT	# 45C5	RIGHT$	# 49D5
SCRN	# 4E46	LEFT$	# 4945	MID$	# 564E
GO	# 5246				

Pourriez-vous expliquer ce phénomène et donner le moyen pour obtenir les bonnes adresses ?

D'autre part, la séquence CALL # EA79 qui simule une coupure d'alimentation, retourne parfois le message OUT OF MEMORY ERROR. Pour remédier, il suffit de taper HIMEM # 9EFF (pour retourner à FRE (Ø) = 39 420) ou HIMEM # 97FF pour éviter une éventuelle reconfiguration des caractères. D'autre part, bien que ce soit une coupure de courant, on garde les couleurs (encre et papier) et le contenu de la mémoire 618 ou # 26A, c'est-à-dire que les indicateurs (ctrl F, ctrl Q, ctrl S, etc ; ;.) restent en l'état.

Enfin, vous serait-il possible de communiquer les adresses d'autres fonctions qui, si elles existent, permettraient, comme ! (#2F5(OU & (# 2FC), des assignations de la forme DOKE # 2F5, adresse.

Merci d'avance,

Voici la réponse proposée par Fabrice BROCHE :

Votre lettre est intéressante car elle nous fournit l'occasion d'apporter des précisions sur le fonctionnement interne de l'**interpréteur.**

L'ensemble des mots-clés BASIC, codés de #8Ø à # F5 peut être divisé en 4 catégories.

① La première, de # 8Ø à # C1, soit de END à NEW est celle des commandes : les maîtres du BASIC.
Leur adresse est effectivement donnée par DEEK ((CODE—128) * 2 + # CØØ6) + 1

② La deuxième catégorie est particulière ; on pourrait l'appeler celle des **attributs de commande :** ce sont les codes de #C2 à #CB de TAB(à STEP. En effet, ces mots clés ne sont rencontrés qu'à l'exécution d'une commande : ainsi TO ou STEP ne sont envisageables qu'après un FOR ; TAB(, SPC(, qu'après un PRINT ou LPRINT. (@ n'est pas traité par la version 1.Ø). C'est la raison pour laquelle ces mots-clés n'ont pas d'adresse dans la table. Des indications erronées ont pu être publiées par exemple dans "VISA pour l'ORIC" ou dans "ESSAI de l'ATMOS" de MICROSYSTÈMES.

③ La troisième catégorie concerne les opérateurs codés de # CC à # D5.Pour ces opérateurs, l'ORIC stocke non seulement l'adresse d'exécution, mais sa priorité : il lui faut, en effet, savoir que * doit être exécuté avant + par exemple.
Vous obtiendrez ce code de priorité par
PEEK (#CØCC + 3*(CODE— #CC))
et l'adresse de l'opération par
DEEK (# CØCD+3*(CODE— #CC)) + 1

④ La quatrième catégorie enfin est celle des fonctions : codes de # D6 à # F6 de SGN à MID$. Les fonctions sont appelées par les routines d'évaluation d'une variable. Vous obtiendrez leur adresse par :
DEEK (CØ8A + 2 * (CODE— #D6))
Voilà pour les codes.

D'autre part, vous avez sans doute remarqué que les ordres sonores et graphiques ont une syntaxe très similaire, c'est la raison pour laquelle ils sont groupés de # A7 à # B2 pour les codes correspondant à SOUD....INK. Ils pointent tous sur un mini-interpréteur qui utilise à son tour une table d'adresses ainsi qu'une table indiquant le nombre des paramètres à utiliser et une troisième table pour le mode d'adressage du curseur en HIRES.

Pour obtenir les adresses :
DEEK (#E84E + 2 * (CODE— #A7)) (V1.Ø)
DEEK (# EAC1 + 2 * (CODE — # A7) (V1.1)

Pour obtenir le nombre de paramètres :
PEEK (# E866 + (CODE — # A7) (V1.Ø)
PEEK (# EAD8 + (CODE — # A7)) (V1.1)

Pour obtenir l'adressage du curseur
PEEK (# E872 + (CODE — # A7)) (V1.Ø)
PEEK (# EAE5 + (CODE — # A7)) (V1.1)

Notez que, pour les routines CURSET, CURMOV etc. l'adresse est un peu antérieure à celle des routines sonores. Un test de mode HIRES est prévu qui envoie le cas échéant un DISP TYPE MISMATCH.

Au sujet de l'exploitation du "!".

Tout d'abord, il ne peut remplacer que des commandes et non des fonctions ni des opérateurs. D'autre part, il se substitue directement au code de la fonction BASIC. Exemple avec l'adresse du HIMEM :
DOKE # 2F5, # E95B (V1:Ø)
DOKE # 2F5, # EBCE (V1.1) ensuite :
! # 97FE équivaut à HIMEM # 97FE.

Ce genre d'utilisation n'est pas possible pour les routines sonores ou graphiques HIRES car elles utilisent, pour les distinguer, le code de la commande.

• A propos de INVERSE et NORMAL

Sur l'ORIC 1, en effet INVERSE et NORMAL sont directement branchées sur la routine d'affichage de "SYNTAX ERROR". Il semble que ces deux mots, initialement prévus aient été abandonnés et remplacés par les procédures connues mais non effacés dans la ROM (V1.Ø).

"!" est branché sur JMP (# 2F5) : saut à l'adresse contenue en # 2F5. A l'initialisation "!" est dérivé vers "ILLEGAL QUANTITY ERROR" alors que "SYNTAX ERROR" conviendrait mieux, s'agissant d'une commande.

• Au sujet des réinitialisations

Il faut distinguer le **"RESET à Froid"** ou vrai RESET du **"RESET à Chaud"** ou NMI, ce dernier étant donné par le bouton sous l'ORIC. Il convient ensuite de distinguer le **"RESET SYSTÈME"** du **"RESET BASIC"** .

Le RESET SYSTÈME remplit la mémoire de "U", code 85 ou # 55, teste si l'on dispose de 16 Ko ou de 48 Ko, initialise le VIA pour le RESET à froid et réinitialisent le VIA pour le RESET à chaud.

Il exécute ensuite un saut vers le RESET BASIC qui affiche "ORIC EXTENDED BASIC V1....", après avoir sélectionné le mode TEXT, initialise certains pointeurs tels !, &, USR, Ready, etc.

Le RESET SYSTÈME se termine par un saut au RESET BASIC. Voici les adresses.

RESET	SYSTÈME	# F42D	# F88F
NMI	SYSTÈME	# 22B	# 247
RESET	BASIC	# CØØØ (ou # EA59)	# CØØØ (ou # ECCC)
NMI	BASIC	# CØØ3 (ou # C475)	# CØØ3 (ou # C471)
		en V1.Ø	en V1.1

Enfin, seule une commande ! en # 2F5 et deux fonctions & en # 2FC et USR (ou DEF USR) en # 22 (par # 21) sont vectorisées en mémoire.

Nous espèrons vous avoir apporté une réponse claire et précise.

Max HAGENBURGER vous fait profiter de ses trouvailles

Quelques remarques à propos de l'utilisation de l'ORIC-1 et de l'ATMOS

— Première remarque : que ce soit ORIC-1, ou ATMOS, le IF imbriqué ne supporte qu'un seul ELSE :

A=TRUE : B=TRUE ou A=FALSE etc.
IF A THEN IF B THEN PRINT "A & B vrais" ELSE "A ou B faux"

— Pour ORIC-1 le ELSE fonctionne pour les 2 conditions A et B comme pour un AND :

IF A AND B THEN PRINT "A et B vrais" ELSE PRINT "A ou B faux"

— pour ATMOS le ELSE ne dépend que du deuxième IF :

IF A THEN IF B THEN PRINT "A & B vrais" ELSE PRINT "A vrais, B faux"

le deuxième cas, celui de l'ATMOS, est plus sélectif et donc plus intéressant et en fait correspond à une structure très riche de la logique élementaire : A et non B

— Puisqu'on parle de logique je voudrais rappeler qu'il est normalement interdit de "sortir" d'un bloc comme GOSUB............RETURN mais aussi de REPEAT .... UNTIL, au risque de perdre les adresses retour des autres blocs ou d'avoir un RETURN WITHOUT GOSUB ERROR....

exemple :
```
10 GOSUB 100 : END
100 REPEAT
110 I=I+1 : PRINT "montant" I :
    INPUT A : TOTAL=TOTAL+A
120 IF A=Ø THEN 14Ø ◄—— interdit!
130 UNTIL I > 1Ø
140 RETURN
```
il faut remplacer les lignes 12Ø et 13Ø par 13Ø UNTIL I > 1Ø OR A=Ø

— le Positionnement du curseur depuis le haut de l'écran : sur ORIC-1 il s'obtient par POKE 616,N où N est le N° de ligne suivi d'un Premier PRINT pour positionner le curseur exemple : CLS : POKE 616,2Ø : PRINT "Pos." PRINT "ligne 1" : PRINT "ligne 2" sur ATMOS, POKE 616,N n'a pour effet que de faire considérer la ligne actuelle comme étant la Nième ligne, ce qui déplace la fenêtre d'écriture par rapport à l'écran mais ne déplace pas le curseur.
essayer : CLS : POKE 616,25 : FOR I=1 TO 2Ø PRINT I : NEXT
surprenant n'est-ce-pas ? il ne reste plus qu'à faire CLS.
Par contre DOKE 18,48ØØØ+N*4Ø a l'effet voulu pour les PRINT suivants mais malheureusement les lignes sont comptabilisées dans 616 inchangé jugez vous-même : CLS : DOKE 18,48ØØØ+25*4Ø FOR I=1 TO 2Ø : PRINT I : NEXT
faites remonter le curseur pour voir l'effet (puis faites CLS).
CLS : DOKE 18,48ØØØ+25*4Ø : POKE : 616,25 : FOR I=1 TO 2Ø : PRINT I : NEXT est plus approprié et DOKE 18,48ØØØ+25*4Ø : POKE 616,25 prend 1/40e de seconde mais malheureusement ce n'est pas utilisable sur ORIC-1 (DOKE 18, ne sert que pour un PRINT) donc les programmes ne seront pas compatibles
Je propose : CLS : FOR I=1 TO 25 : PRINT I : NEXT qui prend 1/12e de seconde

— autre remarque : vous aurez probablement constaté l'erreur de la page 255 de l'ATMOS MANUAL, à la ligne 21Ø du programme "Bogus Error" il manque un # 6Ø   21Ø DATA # 4C, # 22, # EE, # 6Ø.

De plus à la ligne 3Ø, # 28 doit être remplacé par # 281 en conformité avec la ligne 9Ø.

Monsieur P. GUILLERME
nous envoie des documents susceptibles d'intéresser des lecteurs

A la suite du programme de L. AUGUSTONI paru dans le numéro 3 de MICR'ORIC, pourquoi ne pas ajouter les lignes d'un programme permettant d'imprimer les lettres accentuées du français, avec la GP100-A, avec la GP50-A, et aussi, bien entendu avec la MCP-40 ?

Naturellement, il faut ajuster le nombre de caractères par ligne de chaque type d'imprimante :

80 pour la GP1Ø-A,
46 pour la GP 5Ø-A,
4Ø pour la MCP-4Ø.

Pour la GP1ØØ-A, sur laquelle il existe des caractères accentuées possédant des CHR$, ce sont ces caractères qui ont été employés de préférence, mais pour les caractères manquants ou insatisfaisants, on a eu recours à des caractères composés dans le programme.

Il en est ainsi pour le à, le ç, le ê, le ô, le â par exemple.

Pour la GP5Ø-A, où les caractères accentués sont inexistants, on a uniquement recours à des caractères composés en mode graphique.

Pour la MCP-4Ø, sur laquelle déplacements verticaux et juxtapositions sont possibles, les accents sont accolés aux caractères miniscules d'origine.

REMARQUE

Pour faciliter la reconnaissance des minuscules accentuées sur le clavier du microordinateur, on peut avoir recours à des pastilles adhésives de diamètre 9 mm, sur lesquelles on dessine le caractère spécialement créé. Sur l'ORIC-1, une pastille entière peut être collée à côté de ceux qui y figurent déjà. Sur l'ATMOS, il faut se contenter d'une demi-pastille, que l'on arrive, avec un peu d'adresse, à coller sur la touche elle-même sans cacher ce qui est déjà gravé.

Voici ces Programmes :

```
1 REM PREAMBULE COMMUN AUX TROIS IMPRIMANTES
4 DATA@,8,4,28,2,30,34,30,0
5 DATA^,0,0,14,16,16,14,4,8
10 DATA{,28,34,28,34,62,32,30,0
20 DATA[,4,8,28,34,62,32,30,0
30 DATA],16,8,28,34,62,32,30,0
40 DATA},20,0,28,34,62,32,30,0
70 DATA|,8,54,0,34,34,38,26,0
75 DATA*,8,20,0,28,34,34,28,0
80 DATA\,8,54,0,24,8,8,28,0
90 DATA_,8,20,0,30,34,34,30,0
95 DATA&,16,12,0,34,34,38,26,0
100 FOR I=1 TO 11
110 READ Z9$:C=ASC(Z9$)
115 FOR N=0 TO 7:READ B
120 POKE 46080+8*C+N,B
130 NEXT N,I
140 LPRINT
150 DATA"@ ^ { [ ] } | * \ _ &
155 DATA"e [ ] { }
160 DATA"a@£ c^ e[]{} i\ o* u&!
295 DATA"FIN"
298 FOR N=1 TO 100
299 READ A$
300 FOR M=1 TO LEN(A$)
310 IF MID$(A$,M,1)=""THEN LPRINT
311 IF MID$(A$,M,1)=""THEN GOTO 299
315 IF ASC(MID$(A$,M,1))=64 THEN GOTO 1110     '@
320 IF ASC(MID$(A$,M,1))=94 THEN GOTO 1120     '^
330 IF ASC(MID$(A$,M,1))=123 THEN GOTO 1130    '{
```

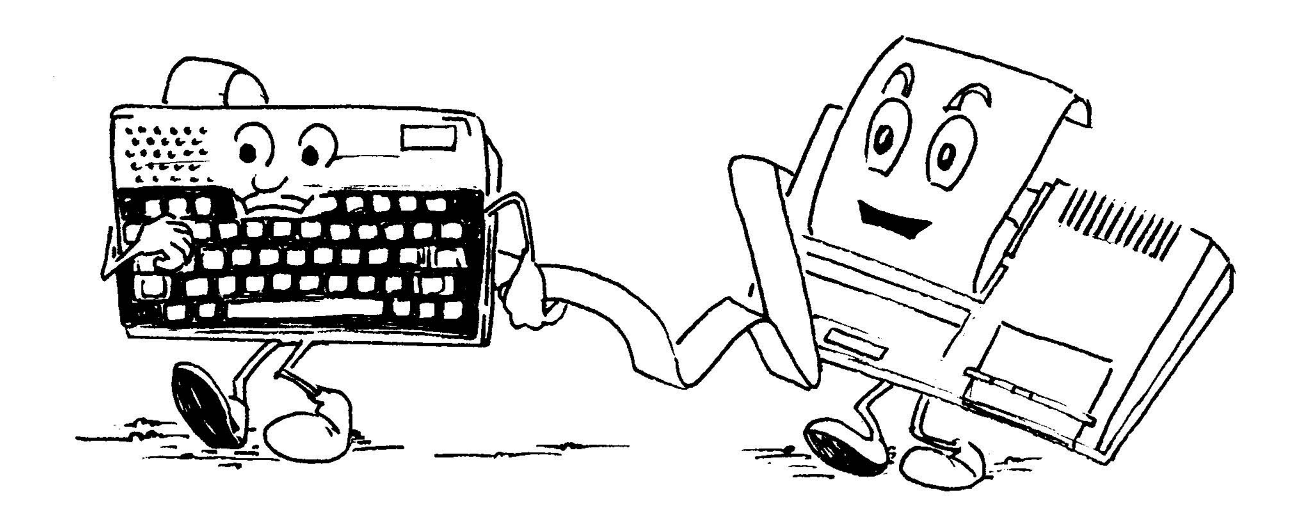
```
340 IF ASC(MID$(A$,M,1))=91 THEN GOTO 1140   '[
350 IF ASC(MID$(A$,M,1))=93 THEN GOTO 1150   ']
360 IF ASC(MID$(A$,M,1))=125 THEN GOTO 1160  '}
370 IF ASC(MID$(A$,M,1))=124 THEN GOTO 1170  '|
380 IF ASC(MID$(A$,M,1))=42 THEN GOTO 1180   '*
390 IF ASC(MID$(A$,M,1))=92 THEN GOTO 1190   '\
400 IF ASC(MID$(A$,M,1))=95 THEN GOTO 1200   '£
410 IF ASC(MID$(A$,M,1))=38 THEN GOTO 1210   '&
415 IF A$="FIN"THEN END
420 LPRINT MID$(A$,M,1);
430 NEXT M
440 LPRINT
450 NEXT N
```

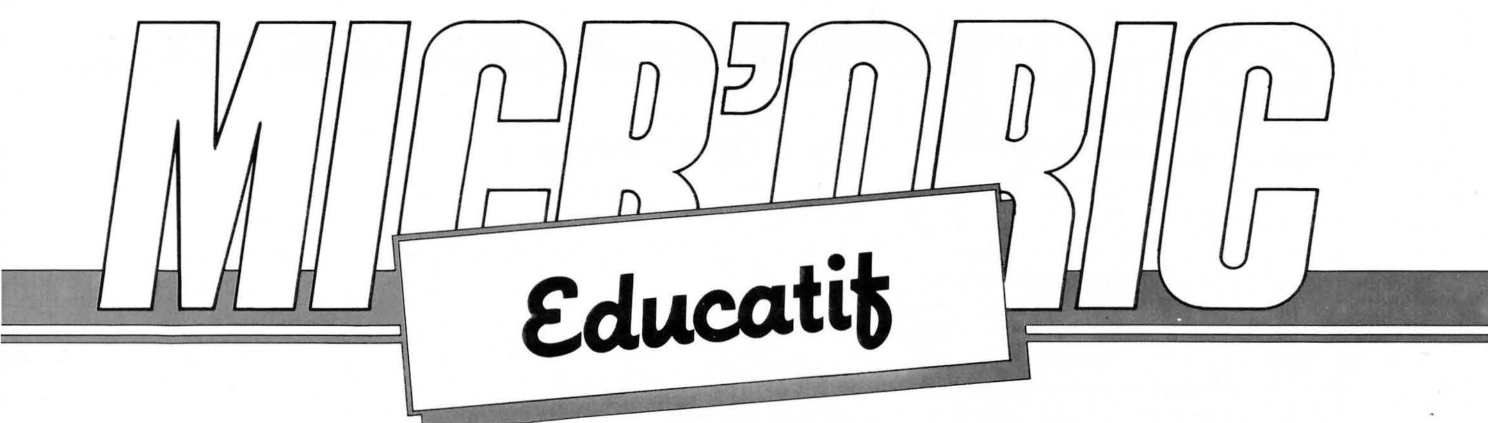
```
1 REM FIN DU PROGRAMME POUR LA GP100-A
1110 LPRINT CHR$(#08);CHR$(128);    '@
1111 LPRINT CHR$(160);CHR$(213);CHR$(214);CHR$(188);CHR$(192);
1112 LPRINT CHR$(#0F);
1113 M=M+1
1114 GOTO 310
1120 LPRINT CHR$(#08);CHR$(128);    '^
1121 LPRINT CHR$(152);CHR$(164);CHR$(228);CHR$(228);CHR$(164);
1122 LPRINT CHR$(#0F);
1123 M=M+1
1124 GOTO 310
1130 LPRINT CHR$(#08);CHR$(128);    '{
1131 LPRINT CHR$(184);CHR$(214);CHR$(213);CHR$(214);CHR$(136);
1132 LPRINT CHR$(#0F);
1133 M=M+1
1134 GOTO 310
1140 LPRINT CHR$(166);     '[
1143 M=M+1
1144 GOTO 310
1150 LPRINT CHR$(167);     ']
1153 M=M+1
1154 GOTO 310
1160 LPRINT CHR$(168);     '}
1163 M=M+1
1164 GOTO 310
1170 LPRINT CHR$(#08);CHR$(128);     '!1171LPRINT CHR$(186);CHR$(193);CHR$(19
1171 LPRINT CHR$(186);CHR$(193);CHR$(193);CHR$(186);CHR$(192);
1172 LPRINT CHR$(#0F);
1173 M=M+1
1174 GOTO 310
1180 LPRINT CHR$(#08);CHR$(128);    '*
1181 LPRINT CHR$(184);CHR$(198);CHR$(197);CHR$(198);CHR$(184);
1182 LPRINT CHR$(#0F);
1183 M=M+1
1184 GOTO 310
1190 LPRINT CHR$(#08);CHR$(128);    '\
1191 LPRINT CHR$(128);CHR$(198);CHR$(253);CHR$(194);CHR$(128);
1192 LPRINT CHR$(#0F);
1193 M=M+1
1194 GOTO 310
1200 LPRINT CHR$(#08);CHR$(128);    '_
1201 LPRINT CHR$(160);CHR$(214);CHR$(213);CHR$(190);CHR$(192);
1202 LPRINT CHR$(#0F);
1203 M=M+1
1204 GOTO 310
1210 LPRINT CHR$(182);     '&
1213 M=M+1
1214 GOTO 310
```

```
1 REM FIN DU PROGRAMME POUR LA GP50-A
 1110 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1111 LPRINT CHR$(32);CHR$(85);CHR$(86);CHR$(6
0);CHR$(64);
 1112 LPRINT CHR$(0);CHR$(0);
 1113 M=M+1
 1114 GOTO 310
 1120 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1121 LPRINT CHR$(24);CHR$(36);CHR$(100);CHR$(
164);CHR$(36);
 1122 LPRINT CHR$(0);CHR$(0);
 1123 M=M+1
 1124 GOTO 310
 1130 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1131 LPRINT CHR$(56);CHR$(86);CHR$(85);CHR$(8
6);CHR$(72);
 1132 LPRINT CHR$(0);CHR$(0);
 1133 M=M+1
 1134 GOTO 310
 1140 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1141 LPRINT CHR$(56);CHR$(84);CHR$(86);CHR$(8
5);CHR$(72);
 1142 LPRINT CHR$(0);CHR$(0);
 1143 M=M+1
 1144 GOTO 310
 1150 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1151 LPRINT CHR$(56);CHR$(85);CHR$(86);CHR$(8
4);CHR$(72);
 1152 LPRINT CHR$(0);CHR$(0);
 1153 M=M+1
 1154 GOTO 310
 1160 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1161 LPRINT CHR$(56);CHR$(85);CHR$(84);CHR$(8
5);CHR$(72);
 1162 LPRINT CHR$(0);CHR$(0);
 1163 M=M+1
 1164 GOTO 310
 1170 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1171 LPRINT CHR$(58);CHR$(65);CHR$(65);CHR$(5
8);CHR$(64);
 1172 LPRINT CHR$(0);CHR$(0);
 1173 M=M+1
 1174 GOTO 310
 1180 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1181 LPRINT CHR$(56);CHR$(70);CHR$(69);CHR$(7
0);CHR$(56);
 1193 M=M+1
 1194 GOTO 310
 1200 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1201 LPRINT CHR$(32);CHR$(86);CHR$(85);CHR$(6
2);CHR$(64);
 1202 LPRINT CHR$(0);CHR$(0);
 1203 M=M+1
 1204 GOTO 310
 1210 LPRINT CHR$(#1B);CHR$(#47);CHR$(0);CHR$(
7);
 1211 LPRINT CHR$(56);CHR$(65);CHR$(66);CHR$(5
6);CHR$(64);
 1212 LPRINT CHR$(0);CHR$(0);
 1213 M=M+1
 1214 GOTO 310
```

```
1 REM FIN DU PROGRAMME POUR LA MCP-40
1110 LPRINT"a";    '@
1111 LPRINT CHR$(8);
1112 LPRINT CHR$(18):LPRINT"R-1,4"
1113 LPRINT CHR$(17);:LPRINT CHR$(96);
1114 LPRINT CHR$(0);:LPRINT CHR$(0);
1115 LPRINT CHR$(18):LPRINT"R1,-4"
1116 LPRINT CHR$(17);
1117 M=M+1
1118 GOTO 310
1120 LPRINT"c";    '^
1121 LPRINT CHR$(8);
1122 LPRINT CHR$(18):LPRINT"R-1,-7"
1123 LPRINT CHR$(17);:LPRINT CHR$(126);
1124 LPRINT CHR$(0);:LPRINT CHR$(0);
1125 LPRINT CHR$(18):LPRINT"R1,7"
1126 LPRINT CHR$(17);
1127 M=M+1
1128 GOTO 310
1130 LPRINT"e";    '{
1131 LPRINT CHR$(8);
1132 LPRINT CHR$(18):LPRINT"R-1,4"
1133 LPRINT CHR$(17);:LPRINT CHR$(94);
1134 LPRINT CHR$(0);:LPRINT CHR$(0);
1135 LPRINT CHR$(18):LPRINT"R1,-4"
1136 LPRINT CHR$(17);
1137 M=M+1
1138 GOTO 310
1140 LPRINT"e";    '[
1141 LPRINT CHR$(8);
1142 LPRINT CHR$(18):LPRINT"R0,4"
1143 LPRINT CHR$(17);:LPRINT CHR$(39);
1144 LPRINT CHR$(0);:LPRINT CHR$(0);
1145 LPRINT CHR$(18):LPRINT"R0,-4"
1146 LPRINT CHR$(17);
1147 M=M+1
1148 GOTO 310
1150 LPRINT"e";    ']
1151 LPRINT CHR$(8);
1152 LPRINT CHR$(18):LPRINT"R-1,4"
1153 LPRINT CHR$(17);:LPRINT CHR$(96);
1154 LPRINT CHR$(0);:LPRINT CHR$(0);
1155 LPRINT CHR$(18):LPRINT"R1,-4"
1156 LPRINT CHR$(17);
1157 M=M+1
1158 GOTO 310
1160 LPRINT"e";    '}
1161 LPRINT CHR$(8);
```

```
1162 LPRINT CHR$(18):LPRINT"R-1,4"
1163 LPRINT CHR$(17);:LPRINT CHR$(34);
1164 LPRINT CHR$(0);:LPRINT CHR$(0);
1165 LPRINT CHR$(18):LPRINT"R1,-4"
1166 LPRINT CHR$(17);
1167 M=M+1
1168 GOTO 310
1170 LPRINT"u";    '!
1171 LPRINT CHR$(8);
1172 LPRINT CHR$(18):LPRINT"R-1,4"
1173 LPRINT CHR$(17);:LPRINT CHR$(94);
1174 LPRINT CHR$(0);:LPRINT CHR$(0);
1175 LPRINT CHR$(18):LPRINT"R1,-4"
1176 LPRINT CHR$(17);
1177 M=M+1
1178 GOTO 310
1180 LPRINT"o";    '*
1181 LPRINT CHR$(8);
1182 LPRINT CHR$(18):LPRINT"R-1,4"
1183 LPRINT CHR$(17);:LPRINT CHR$(94);
1184 LPRINT CHR$(0);:LPRINT CHR$(0);
1185 LPRINT CHR$(18):LPRINT"R1,-4"
1186 LPRINT CHR$(17);
1187 M=M+1
1188 GOTO 310
1190 LPRINT"i";    '\
1191 LPRINT CHR$(8);
1192 LPRINT CHR$(18):LPRINT"R,1"
1193 LPRINT CHR$(17);:LPRINT CHR$(94);
1194 LPRINT CHR$(0);:LPRINT CHR$(0);
1195 LPRINT CHR$(18):LPRINT"R,-1"
1196 LPRINT CHR$(17);
1197 M=M+1
1198 GOTO 310
1200 LPRINT"a";    '
1201 LPRINT CHR$(8);
1202 LPRINT CHR$(18):LPRINT"R-1,4"
1203 LPRINT CHR$(17);:LPRINT CHR$(94);
1204 LPRINT CHR$(0);:LPRINT CHR$(0);
1205 LPRINT CHR$(18):LPRINT"R1,-4"
1206 LPRINT CHR$(17);
1207 M=M+1
1208 GOTO 310
1210 LPRINT"u";    '&
1211 LPRINT CHR$(8);
1212 LPRINT CHR$(18):LPRINT"R-1,4"
1213 LPRINT CHR$(17);:LPRINT CHR$(96);
1214 LPRINT CHR$(0);:LPRINT CHR$(0);
1215 LPRINT CHR$(18):LPRINT"R1,-4"
1216 LPRINT CHR$(17);
1217 M=M+1
1218 GOTO 310
```

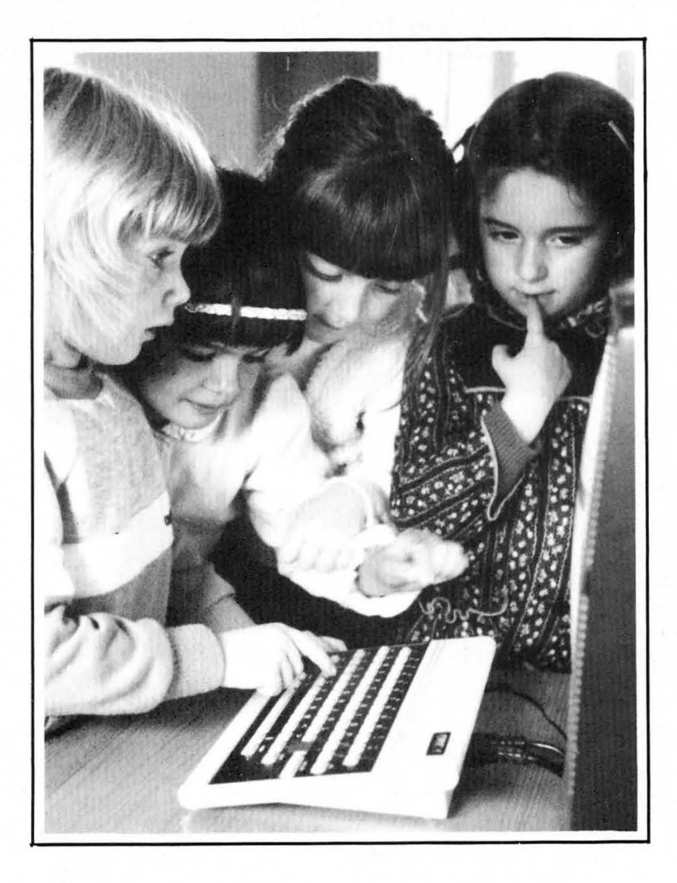




ORIC EN MATERNELLE

par Philippe BRAX

Ce programme a été expérimenté comme en témoigne la photographie.



L'utilisation de ce programme suppose la présence d'un adulte à côté des enfants.

Il s'agit d'un exercice de recherche et de reconnaissance de symboles. Deux modalités : sans appel à la mémoire de l'enfant ou avec nécessité de se souvenir d'un symbole qui reste affiché 3 secondes.

L'enfant doit appuyer sur la touche correspondante.

Une musique récompense la réussite.
Les données sont à adapter à vos besoins ou à vos goûts.

La version que voici est réglée pour tourner aussi bien sur ATMOS que sur ORIC-1. Le premier ZK concerne l'affichage par PLOT et TB la tabulation. Noter l'usage de PEEK (#DØØØ) qui vaut 166 sur ORIC-1.

```
5 REM  ********************
10 REM ** RECONNAISSANCE **
20 REM **       DE       **
30 REM **     PRENOM     **
40 REM ********************
41 IFPEEK(#D000)=166THEN ZK=5:TB=18:GOTO45
42 ZK=8:TB=9 'ATMOS
45 CLS:PAPER5:INK4
50 PLOT 3,6,"EXERCICE DE RECHERCHE ET DE":PLOT3,9,"RECONNAISSANCE DE SYMBOLES"
55 PLOT3,14,"1:SANS MEMORISATION":PLOT3,16,"2:AVEC MEMORISATION":GET A$
56 IF ASC(A$)<49ORASC(A$)>50 THEN 55
60 CLS:PLOT5,8,"L'eleve appuie sur son signe"
65 PLOT 5 ,23,"(SHIFT) à pour changer d'exercice"
70 GET P$:IF ASC(P$)=64 THEN 45
80 IF ASC(P$)<58 AND ASC(P$)>47 THEN      C=21:GOTO 100
90 IF ASC(P$)<91 AND ASC(P$)>64 THEN      C=64 ELSE 70
100 FOR I=1 TO ASC(P$)-C
110 READ S$
120 NEXT
140 PRINTS CHR$(12) :PRINT:PRINT:PRINT     :PRINT
150 PRINT CHR$(4) CHR$(27)  CHR$(83)     CHR$(27) CHR$(74) CHR$(27) "D     "S$
160 PRINT:PRINT:PRINT CHR$(4)
170 REM
180 REM*** CHOIX DES LETTRES ***
190 REM
200 PRINTCHR$(4);CHR$(27);CHR$(81);     CHR$(27);CHR$(74);
210 L=LEN(S$)
215 PRINT TAB(TB)" ";
220 FOR I=1 TO(L/2)+1
230 K$=MID$(S$,(2*I)-1,1) :IF A$="2"THEN WAIT300:R=R+2:PLOTZK+R,5,"?"
232 IF A$="2" THEN PLOTZK+R,6,"?"
240 GET P$:GOSUB 600:IF K$=P$ THEN GOSUB5000 ELSE 290
250 IF A$="2"THEN PLOTZK+R,5,K$:PLOTZK+R,6,K$
260 GOTO 360
290 PRINT CHR$(27);CHR$(8);CHR$(78);P$;
300 PRINT CHR$(27);CHR$(8);CHR$(17);
310 IFA$="2"THENPLOTZK+R,5,K$:PLOTZK+R,6,K$:WAIT300
315 IFA$="2"THENPLOTZK+R,5,"?":PLOTZK+R,6,"?"
320 GET P$:GOSUB 600
325 PRINTCHR$(27);CHR$(17);
330 IF P$=K$ THEN PING ELSE 290
340 PRINT CHR$(27);CHR$(8);CHR$(74);
350 IF A$="2"THEN PLOTZK+R,5,K$:PLOTZK+R,6,K$
360 PRINT P$ SPC(1)
370 NEXT I:R=0
380 PRINT:PRINT
390 PRINTCHR$(4)
395 WAIT 100
400 REM
410 REM *** MELODIE ***
420 REM
430 REPEAT
440 READ N$
450 UNTIL N$="Z"
460 REPEAT
465 PLAY1,0,0,0
470 READ N,D
480 MUSIC 1,3,N,10
490 PLAY 1,0,0,0
500 WAIT D
510 PLAY 0,0,0,0
520 UNTIL D=39
```



ORIC EN MATERNELLE

```
530 RESTORE:CLS:GOTO 60
570 REM
580 REM ** S/PRG PROTECT.CLAV. **
590 REM
600 IF P$=CHR$(8)OR P$=CHR$(9)OR P$=CHR$(10)OR P$=CHR$(11) THEN P$="?"
610 IF P$=CHR$(27)OR P$=CHR$(32)OR P$=     CHR$(127)OR P$=CHR$(16)THENP$="?"
620 IF P$=CHR$(13) THEN P$="?"
625 IFP$=CHR$(3)THENCLS:PRINTCHR$(4):END
630 RETURN
2000 DATA"S T E P H A N I E","C A R O L I N E","C H R I S T E L L E"
2010 DATA"N A T H A L I E","S A N D R I N E","S E V E R I N E"
2020 DATA "V I R G I N I E","B E R T R A N D","S T E P H A N E","E R I C"
2030 DATA "M I C H A E L","J E R O M E","C E C I L E","A U D R E Y"
2040 DATA "F L O R E N C E","A N D Y","C E D R I C","F A B R I C E"
2050 DATA "A U R E L I E","L A U R E N C E","A L E X A N D R E","Y V A N"
2060 DATA "K A R E L L E","I N G R I D","P I E R R E","V E R O N I Q U E"
2070 DATA "A D E L E","P A U L E","H E L E N E","D A V I D","M A R I E"
2080 DATA "J E A N - G A E L","S E B A S T I E N","O L I V I E R"
2090 DATA "Z"
3000 DATA 8,40,8,40,8,40,10,40,12,80,10,60,8,40,12,40,10,40,10,40,8,39
5000 REM S/P NOTE AU HASARD
5010 PLAY1,0,0,0
5020 MUSIC1,3,RND(1)*12+1,8
5030 WAIT50
5040 PLAY0,0,0,0
5050 RETURN
```

VISITE EN TÊTE DE RAM (suite)

Cette page contient répétée ⁄16 fois la programmation du VIA interne de l'Oric.

Les adresses de cette page sont exploitées dès que l'on a des problèmes d'entrée sortie.

Cette page ne sera pas détaillée ici (cela nous mènerait trop loin), mais vous pourrez vous reporter aux livres spécialisés (le manuel de l'ATMOS en parle).

Deux trucs cependant :

La modification de (3Ø6, 3Ø7) permet de changer la vitesse de répétition du clavier (mais perturbe celle du micro-processeur : la vitesse du clavier est inversement proportionnelle à celle du 65Ø2).

L'octet (3Ø2) agit sur le relais du magnéto :
si (3Ø2)=F7 : magnéto éteint
si (3Ø2)=Ø : magnéto allumé
un simple POKE permet d'allumer ou d'éteindre le magnéto.



RESTORE N

ERRATUM page 49 n° 3 dernière ligne remplacer #B5 par # 85 attention N n'est pas une variable mais un n° écrit en chiffres. Exemple !200 met le pointeur de DATA en ligne 200.

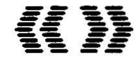
Article : LES VARIABLES

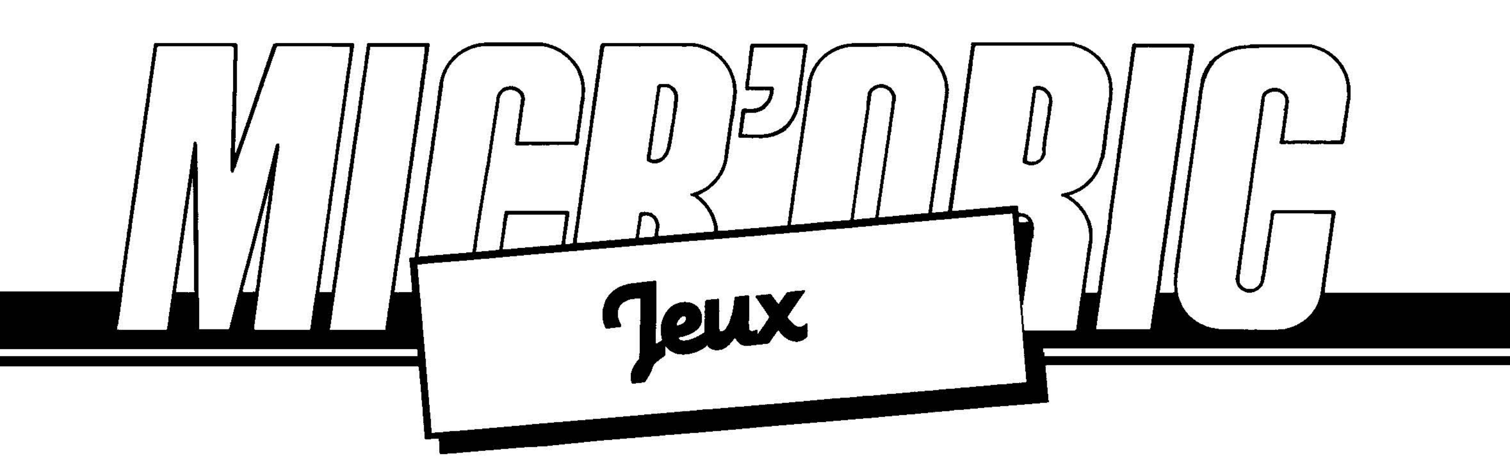
a) page 10 n° 5 en haut 2e ligne l'exposant est négatif 10^{-39} et non 10^{39}. les connaisseurs auront rectifié d'eux-même.

b) page 14 en haut 3 lignes inutiles (voir page 15).

c) le programme page 10 est prévu pour un ORIC 1 sur ATMOS en ligne 700 remplacer # 73 par # 7B, en outre en divers endroits, la précaution MID$ (X$,2) peut être omise ligne 100 par exemple Z$ = STR$ (A−2*x) + Z$.

Le fonctionnement de IF...THEN...ELSE n'étant pas tout à fait le même sur ATMOS il convient également de surveiller l'adaptation. Faire comme si l'on ne disposait pas de ELSE en cas de doute.

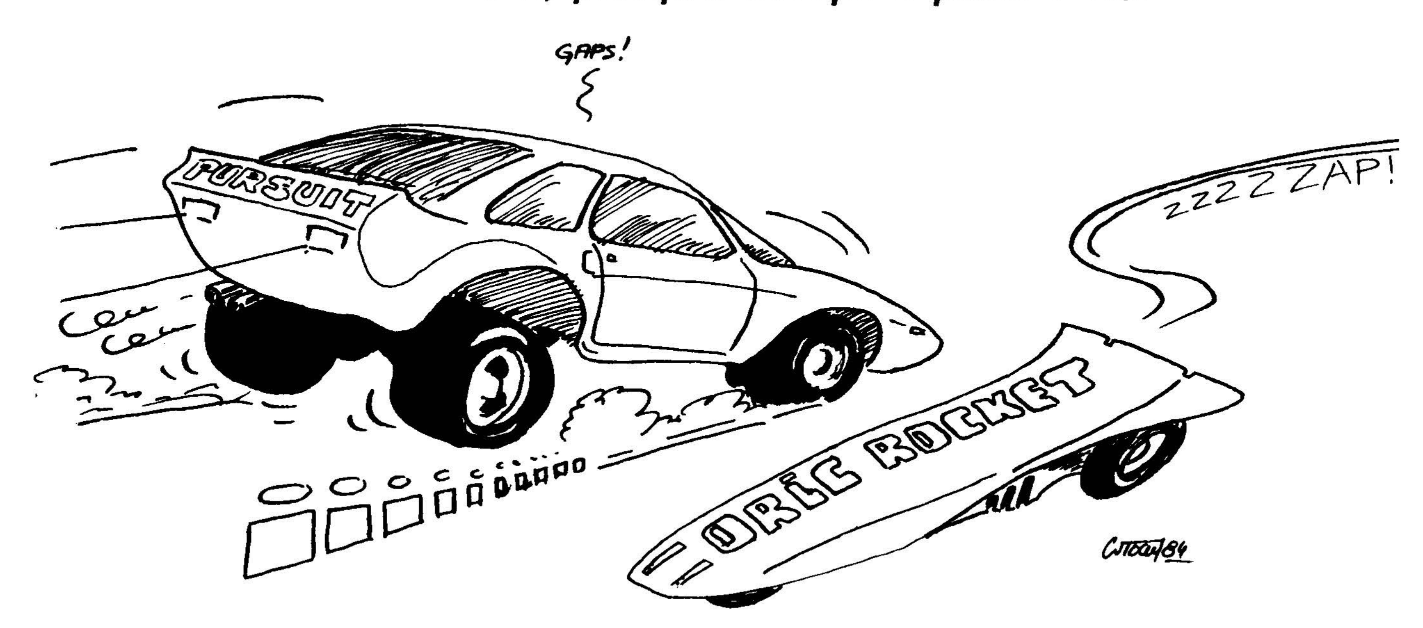




CAR WAR

par Christophe ANDRÉANI

CAR WAR est un jeu d'action, mais aussi de stratégie. Convient à un ORIC 1 16 K ou 48 K, puisqu'il occupe à peine 7 Ko.



Ce logiciel est écrit en BASIC. Il utilise uniquement la page TEXT : on obtient ainsi une rapidité suffisante. Deux mobiles seulement sont animés, mais c'est suffisant pour rendre le jeu intéressant et impératif pour que le BASIC suffise à son affichage dynamique. La vitesse est suffisante : il vous faudra tactique et vivacité pour faire un bon score :

Les premières lignes initialisent l'ORIC : clavier muet, curseur absent, effacement de CAPS, HIMEM, initialisation de l'horloge pour les nombres aléatoires, lecture des DATA pour la reconfiguration des caractères, divers GOSUB pour la présentation et l'implantation de la piste et du décor.

Pour obtenir le plus possible de rapidité, le terrain de jeu a été partagé en 4 quarts : Nord-Ouest, Nord-Est, Sud-Ouest, Sud-Est. Pour chaque zone un sous-programme contrôle les déplacements du véhicule du joueur et comporte les divers tests de collision ou de bonus.

Quand le véhicule quitte un secteur pour aller dans un autre, il y a donc changement de sous-programme. Il en est de même pour "GASPAR" qui est cette chose ronde qu'il faut éviter et qui cherche la collision.

Le véhicule est contrôlé par le clavier, les mouvements de "GASPAR" est programmé en fonction des positions des deux mobiles.

Pour l'ATMOS, remplacer
CALL#E6CA par CALL#E76A
CALL#E8Ø4 par CALL#E93D
CALL#F89B par CALL#F8DØ

CAR WAR

```
10 POKE618,10:POKE524,127
15 FORI=48035TO48039:POKEI,32:NEXT
20 VL=10:W=50
40 GOSUB8000:REM PRESENTATION
60 DOKE#FB,DEEK(#276)
70 DOKE#FD,DEEK(#277)
100 HIMEM#97FF
200 RESTORE
220 FORI=0TO24:READD$:NEXT
250 FORI=46880TO46935:READD:POKEI,D:NEXT
500 CLS:PAPER0
510 INK0
515 C=17:GOSUB20000:REM TERRAIN
516 INK7
520 GOSUB7000:REM BANDES ROUGES
600 REM
610 CALL#E6CA
1000 GOSUB20000:REM TERRAIN
1100 FORI=46880TO46935:READD:POKEI,D:NEXT
1200 V=2
1300 PLOT1,1,102:PLOT3,1,102:PLOT5,1,102
1320 PLOT5,1,32
2000 X1=21:Y1=19
2010 CALL#E804
2020 X2=17:Y2=(INT(RND(9)*5+7))*2+1
2030 CC=1
2035 CALL#E6CA
2040 PLOT33,1,3
2050 VL=10
2100 PLAY1,0,0,0
2500 PLOTX1,Y1,101
2520 PLOTX2,Y2,106
2550 D=1:E=1
2600 PLOT20,12,7:PLOT18,12,CC
2700 GOTO2990
2900 FORI=0TO23:PLOT6,I,RND(9)*7+1:NEXT
2990 REM
3000 ONDGOTO3100,4100,5100,6100
3100 REM
3110 MUSIC1,1,1,0
3120 IFCC=1THENCC=3ELSECC=1
3130 PLOT18,12,CC
3150 IFSCRN(X1+1,Y1)=100THEND=2:GOTO4100
3200 IFSCRN(X1+1,Y1)=105THENSC=SC+10:MUSIC
1,2,8,VL
3250 IFSCRN(X1+1,Y1)=106THEN50000
3270 PLOTX1,Y1,32
3290 CALL#E804
3300 IFX1>17ANDX1<21THEN3400ELSE3500
3400 IFPEEK(#208)=#B0ANDY1>16THENY1=Y1-2
3410 IFPEEK(#208)=#ACANDY1<22THENY1=Y1+2
3420 CALL#E6CA
3500 X1=X1+1
3520 PLOTX1,Y1,101
3550 PLOT34,1,MID$(STR$(SC),2)
3570 IFSC/1000=INT(SC/1000)THEN9000
3800 ONEGOSUB11000,12000,13000,14000
3900 GOTO3100
4100 REM
4110 MUSIC1,1,1,0
4120 IFCC=1THENCC=3ELSECC=1
4130 PLOT18,12,CC
4150 IFSCRN(X1,Y1-1)=100THEND=3:GOTO5100
4200 IFSCRN(X1,Y1-1)=105THENSC=SC+10:MUSIC
1,2,5,VL
4250 IFSCRN(X1,Y1-1)=106THEN50000
4270 PLOTX1,Y1,32
4290 CALL#E804
4300 IFY1>10ANDY1<14THEN4400ELSE4500
4400 IFPEEK(#208)=#9CANDX1>23THENX1=X1-2
4410 IFPEEK(#208)=#BCANDX1<29THENX1=X1+2
4420 CALL#E6CA
4500 Y1=Y1-1
4520 PLOTX1,Y1,102
4550 PLOT34,1,MID$(STR$(SC),2)
4570 IFSC/1000=INT(SC/1000)THEN9000
4800 ONEGOSUB11000,12000,13000,14000
4900 GOTO4100
5100 REM
5110 MUSIC1,1,1,0
5120 IFCC=1THENCC=3ELSECC=1
5130 PLOT18,12,CC
5150 IFSCRN(X1-1,Y1)=100THEND=4:GOTO6100
5200 IFSCRN(X1-1,Y1)=105THENSC=SC+10:MUSIC
1,2,1,VL
5250 IFSCRN(X1-1,Y1)=106THEN50000
5270 PLOTX1,Y1,32
5290 CALL#E804
5300 IFX1>17ANDX1<21THEN5400ELSE5500
5400 IFPEEK(#208)=#B0ANDY1>2THENY1=Y1-2
5410 IFPEEK(#208)=#ACANDY1<8THENY1=Y1+2
5420 CALL#E6CA
5500 X1=X1-1
5520 PLOTX1,Y1,103
5550 PLOT34,1,MID$(STR$(SC),2)
5570 IFSC/1000=INT(SC/1000)THEN9000
5800 ONEGOSUB11000,12000,13000,14000
5900 GOTO5100
6100 REM
6110 MUSIC1,1,1,0
6120 IFCC=1THENCC=3ELSECC=1
6130 PLOT18,12,CC
6150 IFSCRN(X1,Y1+1)=100THEND=1:GOTO3100
6200 IFSCRN(X1,Y1+1)=105THENSC=SC+10:MUSIC
1,2,3,VL
6250 IFSCRN(X1,Y1+1)=106THEN50000
6270 PLOTX1,Y1,32
6290 CALL#E804
6300 IFY1>10ANDY1<14THEN6400ELSE6500
6400 IFPEEK(#208)=#9CANDX1>9THENX1=X1-2
6410 IFPEEK(#208)=#BCANDX1<15THENX1=X1+2
6420 CALL#E6CA
6500 Y1=Y1+1
6520 PLOTX1,Y1,104
6550 PLOT34,1,MID$(STR$(SC),2)
6570 IFSC/1000=INT(SC/1000)THEN9000
6800 ONEGOSUB11000,12000,13000,14000
6900 GOTO6100
```

CAR WAR

```
7000 REM
7010 PLOT6,2,16:PLOT0,2,C:PLOT33,2,C:PLOT5
,4,16:PLOT0,4,C:PLOT34,4,C
7020 PLOT4,6,16:PLOT0,6,C:PLOT35,6,C:PLOT3
,8,16:PLOT0,8,C:PLOT36,8,C
7030 PLOT2,10,16:PLOT0,10,C:PLOT37,10,C:PL
OT1,12,16:PLOT0,12,C:PLOT38,12,C
7040 PLOT1,14,16:PLOT0,14,C:PLOT38,14,C:PL
OT2,16,16:PLOT0,16,C:PLOT37,16,C
7050 PLOT3,18,16:PLOT0,18,C:PLOT36,18,C:PL
OT4,20,16:PLOT0,20,C:PLOT35,20,C
7060 PLOT5,22,16:PLOT0,22,C:PLOT34,22,C:PL
OT6,24,16:PLOT0,24,C:PLOT33,24,C
7070 PLOT0,26,C
7090 RETURN
8000 CLS:PAPER0:INK6
8010 PLOT5,0,"O R I C 1   P R E S E N T E
:"
8015 PLOT4,1,CHR$(5)+"--------------------
---------"
8020 FORI=4TO33STEP2:PLOTI,0,RND(9)*6+1:NE
XT
8030 A$=CHR$(14)+"C A R   W A R"
8040 PLOT10,11,2:PLOT10,12,5
8050 PLOT12,11,A$:PLOT12,12,A$
8060 PLOT1,15,CHR$(1)+"  COPYRIGHT 1984 CH
RISTOPHE ANDREANI"
8062 PLOT2,15,96
8065 PLOT1,16,CHR$(1)+"-------------------
-----------------"
8070 GOSUB8500
8090 PLOT1,25,"VOULEZ VOUS LA LISTE DES IN
STRUCTIONS?"
8100 K$=KEY$
8110 IFK$="o"THEN8200
8120 IFK$="n"THEN8130ELSE8100
8130 RETURN
8200 CLS:INK6:ZAP
8210 PRINT:PRINT"  VOUS COMMANDEZ LE VEHIC
ULE,VOUS"
8220 PRINT:PRINT"DEVEZ EVITER 'GASPARD'.TO
UTE COLLI-"
8230 PRINT:PRINT"SION AVEC 'GASPARD' VOUS
DETRUIT UN"
8240 PRINT:PRINT"VEHICULE.VOTRE BUT EST DE
 RAMASSER"
8250 PRINT:PRINT"LE PLUS DE PASTILLES POSS
IBLE..."
8300 FORI=12TO25:PLOT1,I,3:NEXT
8310 PLOT1,11,17:PLOT1,26,17
8350 PLOT2,13,"      X      : ALLER EN HAU
T"
8360 PLOT2,15,"FLECHE GAUCHE: ALLER EN BAS
"
8370 PLOT2,17,"FLECHE HAUT  : ALLER A GAUC
HE"
8380 PLOT2,19,"FLECHE DROITE: ALLER A DROI
TE"
8400 PLOT7,24,"APPUYEZ SUR UNE TOUCHE"
8410 GETT$
8420 RETURN
8500 PLAY0,0,0,0:PLAY3,0,0,0
8510 GOSUB8900
8520 GOSUB8900
8530 GOSUB8950
8540 GOSUB8900:RETURN
8900 MUSIC2,0,1,VL:MUSIC1,3,8,VL:WAITW:MUS
IC1,3,5,VL:WAITW
8910 MUSIC2,0,1,VL:MUSIC1,3,8,VL:WAITW:MUS
IC1,3,5,VL:WAITW
8920 MUSIC2,0,8,VL:MUSIC1,3,6,VL:WAITW/2:M
USIC1,3,5,VL:WAITW/2:MUSIC1,3,6,VL
8930 WAITW/2:MUSIC1,3,8,VL:WAITW/2:MUSIC2,
0,1,VL:MUSIC1,3,5,VL:WAIT2*W
8940 MUSIC1,1,1,0:MUSIC2,2,2,0:RETURN
8950 MUSIC2,0,8,VL:MUSIC1,3,3,VL:WAITW/2:M
USIC1,1,1,0:MUSIC1,3,3,VL:WAITW/2
8952 MUSIC1,3,6,VL:WAITW/2:MUSIC1,1,1,0:MU
SIC1,3,6,VL:WAITW/2
8955 MUSIC2,0,1,VL:MUSIC1,3,5,VL:WAITW/2
8960 MUSIC1,3,6,VL:WAITW/2:MUSIC1,3,8,VL:W
AITW:MUSIC2,0,8,VL:MUSIC1,3,3,VL
8965 WAITW/2:MUSIC1,1,1,0:MUSIC1,3,3,VL:WA
ITW/2:MUSIC1,3,6,VL:WAITW/2
8967 MUSIC1,1,1,0:MUSIC1,3,6,VL:WAITW/2
8970 MUSIC2,0,1,VL:MUSIC1,3,5,VL:WAITW/2:M
USIC1,3,6,VL:WAITW/2:MUSIC1,3,8,VL
8975 WAITW
8980 MUSIC1,1,1,0
8990 RETURN
9000 REM
9010 VL=10:W=30:GOSUB8500
9050 GOSUB20000:GOTO2000
11000 IFSCRN(X2-1,Y2)=100THENE=2:GOTO12000
11050 IFSCRN(X2-1,Y2)=105THENC=1
11100 IFSCRN(X2-1,Y2)=101THEN50000
11310 IFX2>17ANDX2<21ANDY1>10ANDX2>X1THENP
LOTX2,Y2,32:GOTO11330ELSE11600
11330 IFX1+Y1>=31THEN11500
11340 IFX1+Y2=31THEN11600
11350 IFX1+Y2>31ANDY2>16THENB2=-2ELSEB2=2:
GOTO11600
11500 IFY2-Y1=0THEN11600
11520 IFY2-Y1>0THENB2=-2ELSEB2=2:GOTO11600
11600 IFC=2THENPLOTX2,Y2,105ELSEPLOTX2,Y2,
32
11620 X2=X2-1:Y2=Y2+B2:B2=0
11650 PLOTX2,Y2,106
11700 IFC=1THENC=2ELSEC=0
11750 RETURN
12000 IFSCRN(X2,Y2-1)=100THENE=3:GOTO13000
12050 IFSCRN(X2,Y2-1)=105THENC=1
12100 IFSCRN(X2,Y2-1)=104THEN50000
12310 IFY2>10ANDY2<14ANDX1<21ANDY2>Y1THENP
LOTX2,Y2,32:GOTO12330ELSE12600
12330 IFX1-Y1<=7THEN12500
12340 IFX2-Y1=7THEN12600
12350 IFX2-Y1<7ANDX2<15THENA2=2ELSEA2=-2:G
```

CAR WAR

```
OTO12600
12500 IFX2-X1=0THEN12600
12520 IFX2-X1>0THENA2=-2ELSEA2=2:GOTO12600
12600 IFC=2THENPLOTX2,Y2,105ELSEPLOTX2,Y2,
32
12620 Y2=Y2-1:X2=X2+A2:A2=0
12650 PLOTX2,Y2,106
12700 IFC=1THENC=2ELSEC=0
12750 RETURN
13000 IFSCRN(X2+1,Y2)=100THENE=4:GOTO14000
13050 IFSCRN(X2+1,Y2)=105THENC=1
13100 IFSCRN(X2+1,Y2)=103THEN50000
13310 IFX2>17ANDX2<21ANDY1<14ANDX2<X1THENP
LOTX2,Y2,32:GOTO13330ELSE13600
13330 IFX1+Y1<=31THEN13500
13340 IFX1+Y2=31THEN13600
13350 IFX1+Y2<31ANDY2<8THENB2=2ELSEB2=-2:G
OTO13600
13500 IFY2-Y1=0THEN13600
13520 IFY2-Y1>0THENB2=-2ELSEB2=2:GOTO13600
13600 IFC=2THENPLOTX2,Y2,105ELSEPLOTX2,Y2,
32
13620 X2=X2+1:Y2=Y2+B2:B2=0
13650 PLOTX2,Y2,106
13700 IFC=1THENC=2ELSEC=0
13750 RETURN
14000 IFSCRN(X2,Y2+1)=100THENE=1:GOTO11000
14050 IFSCRN(X2,Y2+1)=105THENC=1
14100 IFSCRN(X2,Y2+1)=102THEN50000
14310 IFY2>10ANDY2<14ANDX1>17ANDY2<Y1THENP
LOTX2,Y2,32:GOTO14330ELSE14600
14330 IFX1-Y1>=7THEN14500
14340 IFX2-Y1=7THEN14600
14350 IFX2-Y1>7ANDX2>23THENA2=-2ELSEA2=2:G
OTO14600
14500 IFX2-X1=0THEN14600
14520 IFX2-X1>0THENA2=-2ELSEA2=2:GOTO14600
14600 IFC=2THENPLOTX2,Y2,105ELSEPLOTX2,Y2,
32
14620 Y2=Y2+1:X2=X2+A2:A2=0
14650 PLOTX2,Y2,106
14700 IFC=1THENC=2ELSEC=0
14750 RETURN
20000 REM
20400 RESTORE
21000 FORI=0TO24
21050 READD$
21100 PLOT7,I,D$
21200 NEXT
25000 RETURN

30000 DATAddddddddddddddddddddddddd,di i i i i     i i i i id
30010 DATAd dddddddd     dddddddd d,didi i i i     i i i idid
30020 DATAd d dddddd     dddddd d d,dididi i i     i i ididid
30030 DATAd d d dddd     dddd d d d,didididi i     i idididid
30040 DATAd d d d dd     dd d d d d,dididididi     ididididid
30050 DATAd         ddddd         d,d         d   d         d
30055 DATAd         d j d         d
30060 DATAd         d   d         d,d         ddddd         d
30070 DATAdididididi     ididididid,d d d d dd     dd d d d d
30080 DATAdidididi i     i idididid,d d d dddd     dddd d d d
30090 DATAdididi i i     i i ididid,d d dddddd     dddddd d d
30100 DATAdidi i i i     i i i idid,d dddddddd     dddddddd d
30110 DATAdi i i i i     i i i i id,ddddddddddddddddddddddddd
```

```
40000 DATA63,51,33,33,33,33,51,63,0,59,16,
31,31,16,59,0,0,45,45,12,45,63,33,0
40010 DATA0,55,2,62,62,2,55,0,0,33,63,45,1
2,45,45,0,0,0,0,12,12,0,0,0
40020 DATA30,51,33,63,63,33,51,30
50000 REM
50010 EXPLODE
50020 PLOTX1,Y1,32
51000 FORI=1TO1000:NEXTI
51010 IFV=0THEN52000
51020 IFV=1THENV=0:PLOT1,1,32
51030 IFV=2THENV=1:PLOT3,1,32
51060 IFC=2THENPLOTX2,Y2,105ELSEPLOTX2,Y2,
32
51080 GOTO2000
52000 REM
52500 CALL#E804
52510 PLOT9,25,"VOULEZ VOUS REJOUER ?"
52515 LU=1
52520 K$=KEY$
52530 IFK$="o"THENCALL#E6CA:GOTO52700
52535 IFK$="n"THEN53000
52540 PLOT8,25,LU
52550 LU=LU+1:IFLU=8THENLU=1
52600 GOTO52520
52700 PLOT9,25,"                             "
52710 SC=0:PLOT34,1,"      "
52720 DE=1:RUN600
53000 CALL#E804
53010 CLS:POKE#20C,255:POKE618,3
53020 CALL#F89B
53050 RUN
```



ORIC MAN

de Marc BELLŒIL

Des jeux d'adresse dans un labyrinthe, vous en connaissez tous. Sur ce sujet voici une proposition de Marc Bellœil qui nous a déjà fourni un mur de briques en langage machine.

Le programme est fourni ici en version BASIC pour ORIC 1, les routines en langage machine sont créées à partir des DATA. Attention en les recopiant !

Pour adapter ce jeu sur ATMOS ajouter en ligne 1Ø :
DOKE#9485,#FACB : DOKE#9731,#CCCE : DOKE#97C6,#FB14
En ligne 34Ø remplacer X=9+Z par X=1Ø+Z
En ligne 36Ø remplacer A=38−X par A=4Ø=X

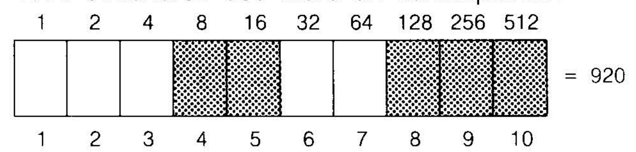
Le jeu est simple, les mouvements de votre personnage "ORIC MAN" sont obtenus avec les flèches du clavier. Il faut passer partout sans se faire attaquer par les monstres dont le déplacement est contrôlé par le programme. Certaines cases vous immunisent et vous effacez une zone plus large lors de vos déplacements. Si vous êtes coincé, il suffit d'appuyer sur @ pour revenir au BASIC.

Vous pouvez choisir le temps de jeu et le niveau de difficulté. Quatre tableaux successifs vous seront proposés.

Lors du premier passage, vous verrez s'afficher les murs du labyrinthe. Par la suite ils apparaîtront d'un coup.

Marc Bellœil vous propose ici une méthode de création du décor intéressante. Voyez les lignes 47Ø à 54Ø. Pour chaque tableau vous avez 2 groupes de 14 données. Conservez 1Ø23 au début.

Ensuite, écrivez des valeurs variées de Ø à 1Ø23 vous obtiendrez des murs en conséquence.

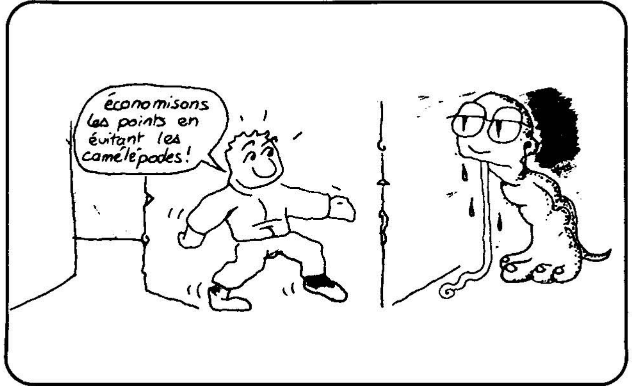


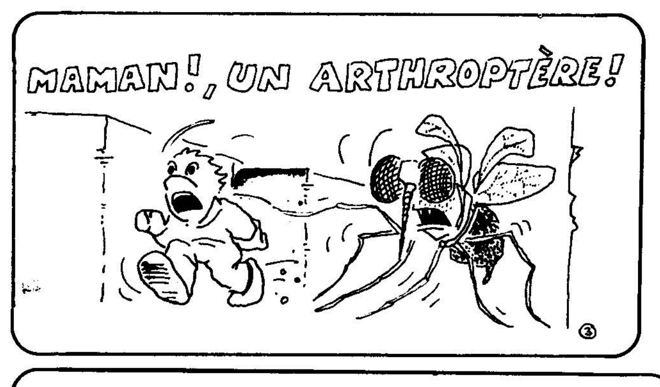
En partant d'en haut à gauche et sur dix cases d'un coup, chaque nombre fait s'afficher un groupe de case colorée sur la ligne correspondante. Vous atteignez ainsi les 14 premières lignes. Simultanément se dessinent les 3 autres coins par symétrie.
Le groupe des 14 autres données correspond à la zone centrale. Veillez à respecter l'accés aux cases occupées par les monstres, l'ORIC-MAN et les cases immunisantes et dopantes.

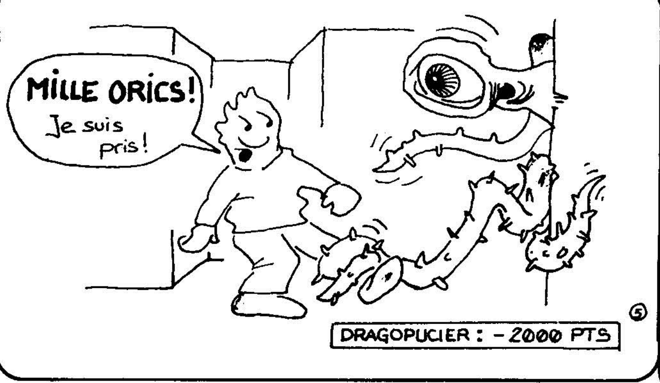
Vous pourrez choisir vous-même vos labyrinthes et ainsi personnaliser ce jeu.

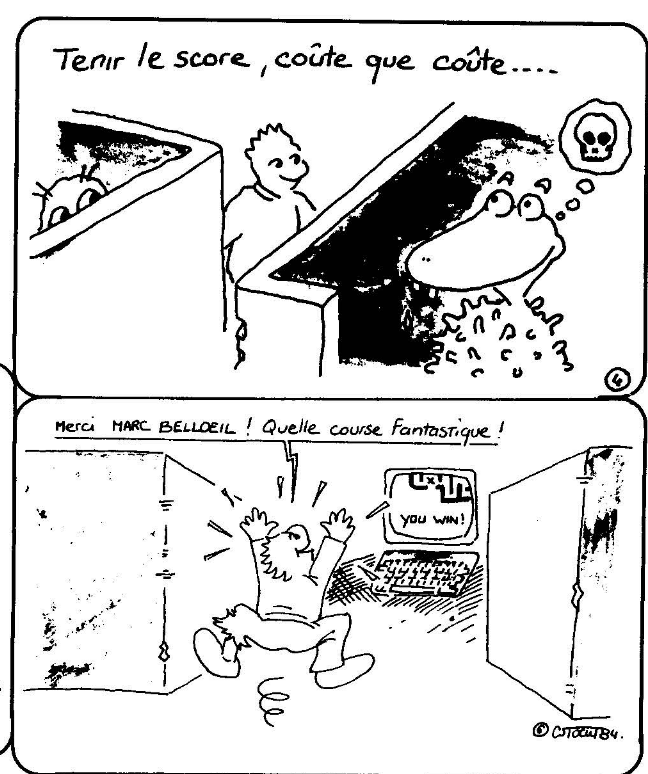
Le tableau récapitulatif des scores est prévu. Les divers affichages sont dynamiques et bruités de façon plaisante. Notre dessinateur Gilles Tocut exprime avec humour ce qu'on peut imaginer en jouant. Bon amusement.












```
1 TEXT:HIMEM #6000:INK6:PAPER0:CLEAR
5 CLS:POKE 618,10:PRINT:PRINT:PLOT 12,2,"UN INSTANT..."
10 GOSUB 1000:GOSUB 700:CLS
11 PLAY 1,0,1,5000:MUSIC 1,1,1,0
12 PLOT 10,5,CHR$(7)+CHR$(14)+"ORIC-MAN":PLOT 10,6,CHR$(7)+CHR$(14)+"ORIC-MAN"
15 PLOT 5,10,"Pendant combien de secondes"
16 A=0:B=0:C=9:D=14
18 PLOT 7,11,"voulez-vous jouer"
19 PLOT10,12,CHR$(3)+"(5 a 999) :"
20 PLOT 8,14,CHR$(1)+"--- secondes"
21 FORX=1TO3:GET A$:GOSUB 6000
22 IF B=0 AND A$<>"0" THEN GOSUB 6000:A$=" ":GOTO24
23 A=A*10+B
24 PLOT C,D,A$:C=C+1
25 NEXT X
26 IF A<5 THEN 11 ELSE TEM=A
30 PLOT 1,20,CHR$(12)+CHR$(5)+"Votre force (0:maxi a 9:mini) ?"
35 GET A$:GOSUB 6000
40 A=B*8+15
50 POKE #96F0,A:DOKE #12,#BB80:PRINT"Joueur :  "
55 REM
60 FORX=#BB8BTO#BB8F:POKE X,48:NEXT
70 EC=1
80 RESTORE:B$="ECRAN No "+CHR$(48+EC)
85 CLS:PLOT10,12,CHR$(EC+4)+B$
86 WAIT 50
90 READ A$:IF A$=B$ THEN GOSUB 300 ELSE 90
95 DOKE #BDC3,#7D0C:POKE #BDC5,8:A$=KEY$:GETA$:POKE #BDC3,123:POKE #BDC5,123
96 POKE #2DF,ASC(A$)+128
100 POKE #FF,0:CALL#9580
110 IF PEEK(#FF)=#42 THEN FORB=1TO10:GOSUB6020:NEXT:EC=EC-1
120 EC=EC+1:WAIT 50:IF EC<5 THEN 80
130 CLS:PAPER0
132 WAIT 50:A$=KEY$
135 PLOT 1,3,CHR$(1)+"Votre nom :"
140 N$="":PLOT 1,5,CHR$(6)+"--------"
```

```
145 B=0:C=0:D=2
150 FORX=1TO8
153 IF C=0THENGETA$:B=B+1:GOSUB 6020
155 IF ASC(A$)<31 OR ASC(A$)>125 THEN A$=" ":C=1
157 N$=N$+A$:PLOTD,5,A$:D=D+1:NEXTX
160 A$="":FORX=#BB8B TO #BB90:A$=A$+CHR$(PEEK(X)):NEXT
170 A=VAL(A$)
175 C=9:D=10
180 ZZ=1:FOR X=0 TO 8
185 L$=RIGHT$(STR$(X+1),1)
190 IF A>=A(X) THEN A(9)=A(X):N$(9)=N$(X):A(X)=A:N$(X)=N$:A=A(9):N$=N$(9)
193 L$=L$+"   "+N$(X)+"   "+STR$(A(X))
195 IF N$(X)="" THEN 205
196 IF X=0 THEN PLOT C-1,D-1,CHR$(#E)+CHR$(1)+L$
197 IF X=0 THEN PLOT C-1,D,CHR$(#E)+CHR$(1)+L$:GOTO 204
200 PLOT C,D,CHR$(INT(6*X/8)+1)+L$
204 D=D+1
205 NEXT
206 IF N$<>"" THEN PLOT 2,D,N$+" n'est pas classe(e) ":D=D+1
207 WAIT 100:A$=KEY$
210 PLOT 1,D+3,"Une autre partie ? "
220 GETA$:IFA$="O" THEN POKE 618,10:CLS:GOTO11
230 IF A$<>"N" THEN CLS:PRINT"(repondez par oui:O ou par non:N )";:GOTO 220
240 END
280 REM ------------------------------
290 REM
300 REM          Ecrans
304 REM
305 REM ------------------------------
310 CLS:Z=0:Y=0
320 IF PEEK(#6001)=#FF THEN 4100 ELSE CALL#9520:GOSUB 5000
330 READ N
340 X=9+Z:U=512
350 REPEAT:IF N<U THEN 400
360 A=38-X:B=26-Y
370 PLOT A,Y,255:PLOT A,B,255
380 PLOT X,Y,255:PLOT X,B,255
390 N=N-U
400 X=X-1:U=U/2:UNTIL X=Z-1
410 IF Y=13 AND Z=10 THEN 435
420 IF Y=13 THEN Z=10:Y=-1
430 Y=Y+1:GOTO 330
435 A=124:POKE#BEDC,A:POKE #BCAC,A:POKE #BC9B,A:POKE #BCBD,A
436 IF EC=4 THEN X=9:Y=5
440 IF EC=1THENX=12:Y=7
441 IF EC=2THENX=9:Y=7
442 IF EC=3 THEN X=5:Y=10
443 A=38-X:B=26-Y
444 PLOT X,Y,126:PLOT A,B,126:PLOT X,B,126:PLOT A,Y,126:POKE 6,0
446 GOTO 4030
450 DOKE #FA,#BBA8:POKE #FE,1:POKE #FD,0:CALL#9540:RETURN
470 DATA ECRAN No 1
471 DATA 1023,17,97,397,529,33,801,145,73,569,257,385,249,1,1023,4,36,192,143
472 DATA 144,160,33,146,140,64,32,30,0
490 DATA ECRAN No 2
500 DATA 1023,1,61,129,249,9,9,9,9,41,41,993,7,17,1023,0,240,4,124,64,64,64
505 DATA 64,80,80,543,0,0
510 DATA ECRAN No 3
511 DATA 1023,1,253,129,129,129,129,129,129,253,1,253,5,5
512 DATA 1023,0,511,257,257,287,272,272,848,80,80,991,0,0
530 DATA ECRAN No 4
535 DATA 1023,1,1,1,9,17,33,97,17,777,13,785,289,33
540 DATA 1023,0,131,68,40,16,8,136,132,99,529,304,192,0
680 REM ------------------------------
690 REM
700 REM        Caracteres
701 REM
```

```
702 REM ------------------------------
705 F=0:RESTORE
710 READ A$:IF A$="CARACTERES" THEN 720 ELSE 710
720 REPEAT
730 READ A:POKE #B400+123*8+F,A
735 F=F+1
740 UNTIL F=32
750 RETURN
790 DATA CARACTERES
800 DATA 0,0,0,24,24,0,0,0,#1E,#3F,#2D,#3F,#12,#1E,#C,#3F,#C,#12,#3F,#3E,#3C
810 DATA #3F,#1E,#C
820 DATA12,18,45,45,18,12,0,0
900 REM ------------------------------
910 REM       Implantation
1000 REM           du
1001 REM   programme machine
1002 REM ----------------------------
1010 READ A$:IF A$<>"PROGRAMME" THEN 1010
1015 INK2:TT=30:M1=1
1020 A=#9474:READ T$
1030 FOR X=1 TO LEN(T$)-1 STEP 2
1040 B$=MID$(T$,X,2):IF B$="  " THEN 1055
1045 V=VAL("#"+B$):POKE A,V
1050 A=A+1
1055 NEXT
1060 READ T$:TT=TT-1:PLOT 15,3,"-"+STR$(TT)+" - "
1070 M0=M0+1
1075 IF M0>12 THEN M0=1:M1=M1+1
1076 MUSIC 1,M1,M0,0:PLAY 1,0,1,2500
1080 IF T$<>"FINI"THEN 1030
1090 RETURN
1100 DATA PROGRAMME
1105 DATA A97D91F4ADDF02F0FB4CEE95
1110 DATA A506D00420B1FA60A90085068DDF02A000A92091F4A9C485F4A9BD85F54C7494
1200 DATA C97ED004A9508506A506F024C606A5F4A6F538E928B001CA85038604A05020D7
1300 DATA 94A02720D794A02920D794A00020D794A97D91F44C2F96EAB103C97BF00160A9
1400 DATA 209103A205209697C6FDD002C6FE60EA
2001 DATA A000B1FA91F888D0F9E6FBE6F9A5FBC9C0D0ED60EAA000B1F891FA88D0F9E6FB
2002 DATA E6F9A5FBC9C0D0ED60EAEAEAEAEAEAEA
2003 DATA A97BA00099A8BB99A8BC99A8BD99A8BE99E0BE88D0EEEAEAEAEAEAEAEAEAEAEA
2006 DATA A000B1FAC97BD006E6FDD002E6FEE6FAD002E6FBA5FBC9C0D0E6EAEAEAEAEA60
2010 DATA A5601869289002E66138F162B002C66185609E662D0034CD3964C9D96EAEAEAEA
2020 DATA A2C4A0BD86F484F5A97B8567856B856F8573A2BEA0DC86728471A2BCA0AC8666
2030 DATA 8465A09B866A8469A0BD866E846DEAEAA228A000ADDF02C9C0D0034C3097C988
2040 DATA D002A227C989D002A229C98BD002A200C98AD002A2508600A92091F4A5F438E9
2050 DATA 28B002C6F518650085F49002E6F5B1F4C97CD0044C8094EAC97BD00620E3944C
2060 DATA 2B96EAEAEAEAA6FED0034CCB97C9FFF0034C2B96A9008DDF02A5F41869289002
2070 DATA E6F538E50085F4B002C6F5EA4CA094EAA2978663A2108601A50138E9048501AA
2080 DATA B5658560B5668561B5678564EAEAEAEAA5649160A5F4A6F538E560B001CA8510
2090 DATA 8A38E5618511B013E611F004A0FCD008A510C9E090F6A0F84C8B96A511F004A0
2100 DATA F4D008A510C92090F6A0F08462A601B56829FCC562D004B5688562A000A56038
2110 DATA E928B002C6611871629002E6618560EAB160C97DD0044C8094EAC97CD0034C60
2120 DATA 95C9FFF0F9EAEAEAB1608564A97C9160EAEAEAA601A5609565A5619566A56295
2130 DATA 68A5649567E000F0034C3896EAEAEAA030A2FFCAD0FD4C569788D0F5A010A506
2140 DATA F01EAEE0B7E01ED0044CB095EAEAEAEAEAEAEAEAB9CF9799DFB788D0F74CB095
2150 DATA B9DF9799DFB788D0F7F0F2EAEAEAEAEA200ACCA014B9409799C0BC88D0F760EA
2160 DATA 206175746575722 03A204D2E42454C4C4F45494C20EA8403A5FB38ED7602D005
2170 DATA A4034CF996AD760285FBA00888F01CB99CBBC920F0F6C930F009AACA8A999CBB
2180 DATA 4C5897A939999CBB4C6C97C939D00360604C5897EAA9088D81BBA00588B98B
2190 DATA BB38E93085FA8AF81865FAD885FA290F186930998BBBA5FA29F0F009C000F005
2200 DATA A2014C9D9720FAFAA00060A24286FF601E3F2D3F121E0C3F0C123F3E3C3F1E0C
2210 DATA 3333000C1E211E0C1E3F2D3F121E212150002750295029002729002900270029
2300 DATA FINI
4030 ADD=#8000-(EC-1)*#501
4040 DOKE #F8,ADD:DOKE #FA,#BBA8
4050 CALL #94F0:IF EC=4 THEN POKE #6001,255
4060 GOTO 450
```

(suite page 25)

SOLITAIRE EN RECTANGLE

par Hervé LACOMBE

Tout le monde connait le jeu du SOLITAIRE...

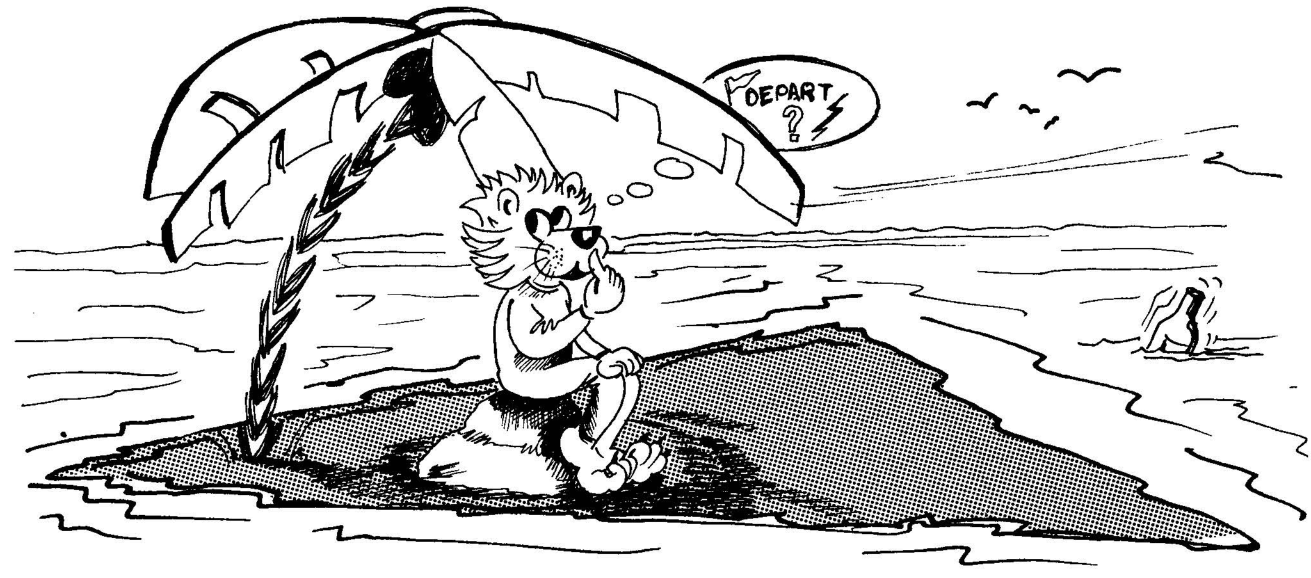
Ici, la piste est rectangulaire et comporte 48 pions ou dames repérées par leur numéro. Pour jouer, il vous suffit d'indiquer la case de départ et la case d'arrivée de la dame que vous voulez déplacer. ORIC s'occupe du reste, compte les points, modifie l'affichage et surveille vos erreurs volontaires ou non. Il n'est pas possible de tricher !

Ce programme tourne sur ORIC 1 ou ATMOS. Seules les lignes 1000 et 1010 sont à changer, sur ORIC 1 celles-ci sont préférables pour le contrôle de l'affichage des INPUT.

```
1000 PRINTCHR$(30):FORI=1TO7:FORJ=2TO4:PLOTI,J,32:NEXTJ:NEXTI
1010 FORI=1TO5 :FORJ=1TO22:PLOTI,J,32:NEXTJ:NEXTI
```

```
15 REM INITIALISATION
50 CLS:PAPER6:INK0:PRINTCHR$(17):FORI=#BBA4TO#BBA7:POKEI,32:NEXTI
60 PRINT:PRINT
80 :DIMA(64):DIMA$(64,2)
100 PRINT"        DAMES EN SOLITAIRE  "
200 PRINT
220 PRINT
300 PRINT" 48 DAMES SONT PLACEES SUR LES"
310 PRINT"2 RANGS EXTERIEURS D'UN DAMIER"
320 PRINT"DE 64 CASES. LE BUT EST D'EN"
330 PRINT"MANGER LE PLUS POSSIBLE."
335 PRINT:PRINT:PRINT"LES DAMES SONT EN NOIR SUR FOND BLEU"
336 PRINT"CIEL. LES CASES VIDES, EN BLANC SUR"
337 PRINT"FOND ROUGE."
340 PRINT:PRINT:PRINT
350 PRINT"VOUS NE POUVEZ SAUTER QU'UNE DAME A LA FOIS ET SEULEMENT EN DIAGONALE
"
360 PRINT:PRINT:PRINT  "POUR COMMENCER, TAPER N'IMPORTE QUELLE TOUCHE"
370 PRINTCHR$(17)
450 IFKEY$=""THENGOTO450
500 PRINTCHR$(17):CLS:PRINT:PRINT
550 REM POUR ECRIRE EN BAS DE L'ECRAN
600 FORI=1TO22:PRINT:NEXT
610 PRINTCHR$(27)"STAPER 1000 QUAND VOUS ETES BATTUS"
630 PRINT:PRINTCHR$(27)"SEN CAS D'ERREUR, VOUS ENTENDREZ PING"
640 REM AFFICHAGE DU DAMIER
650 FORI=6TO30STEP3:FORJ=6TO20STEP2:PLOTI,J,128:NEXTJ:NEXTI
660 FORI=6TO30:FORJ=5TO21STEP2:PLOTI,J,128:NEXTJ:NEXTI
700 FORI=1TO64:B$=STR$(I):X$=RIGHT$(B$,LEN(B$)-1)
702 IFLEN(X$)=1THENX$="0"+X$
710 A$(I,2)=X$:NEXTI:K=0
715 REM AFFICHAGE DES NOMBRES
```

```
720 FORI=7TO28STEP3:FORJ=6TO20STEP2:K=K+1:PLOTI,J,A$(K,2):NEXTJ:NEXTI
730 FORM=19TO43STEP8:FORG=MTOM+3:GOSUB8000:GOSUB9000:NEXTG:NEXTM:GOTO800
800 FORJ=1TO64:A(J)=1:NEXTJ:FORJ=19TO43STEP8:FORI=JTOJ+3:A(I)=0:NEXTI:NEXTJ
960 M=0
1000 PRINTCHR$(30):FORI=2TO7:FORJ=1TO4:PLOTI,J,32:NEXTJ:NEXTI
1010 FORI=2TO5 :FORJ=1TO22:PLOTI,J,32:NEXTJ:NEXTI
1020 PRINT"SAUT  SCORE         MAXI=47":PRINTCHR$(17)
1025 REM SAISIE DU DEPLACEMENT
1030 INPUT F:IFF=1000THENGOTO5000
1100 INPUT T
1150 REM CONTROLE
1200 X=INT((F-1)/8):Y=F-8*X:Z=INT((T-1)/8):W=T-8*Z
1600 IFX>7ORZ>7ORY>8ORW>8ORABS(X-Z)<>2ORABS(Y-W)<>2THENGOTO2300
1610 IFA((T+F)/2)=0ORA(F)=0ORA(T)=1THENGOTO2300
1700 GOTO2450
2300 PING
2350 WAIT 50:PRINTCHR$(17):GOTO1000
2400 REM MISE A JOUR DES TABLEAUX
2450 A(T)=1:A(F)=0:A((T+F)/2)=0
2500 G=T:GOSUB8000:C=C-128:E=E-128:GOSUB9000
2600 G=F:GOSUB8000:GOSUB9000
2700 G=(T+F)/2:GOSUB8000:GOSUB9000
2950 M=M+1:G=M:GOSUB8000:PLOT 8,3,CHR$(E):PLOT 9,3,CHR$(C)
3000 PRINTCHR$(17):GOTO1000
5000 S=0:FORI=1TO64:S=S+A(I):NEXTI:CLS
5600 PRINT
6000 PRINT:PRINT:PRINT"  VOULEZ-VOUS REJOUER ? (O/N)"
6100 INPUTG$:IFG$="O"THENGOTO500
7000 END
8000 REM CARACTERE EN VIDEO INVERSEE
8100 G$=STR$(G):G$=RIGHT$(G$,LEN(G$)-1):IFLEN(G$)=1THENG$="0"+G$
8200 E=ASC(G$)+128
8300 X$=RIGHT$(G$,LEN(G$)-1):C=ASC(X$)+128:RETURN
9000 REM  PLACER UN POINT SUR LE DAMIER
9200 H=INT((G-1)/8):K=G-8*H-1: K=4+2*(K+1):H=7+3*H
9300 PLOTH,K,CHR$(E):PLOTH+1,K,CHR$(C):RETURN
```



PERIPH

9 PERIPHERIQUES

LIAISONS PARTIC

MONITEUR COULEUR NOVEX : La visualisation idéale de votre ordinateur
Le moniteur le plus robuste, le pus fiable et le plus performant pour son prix. Affichage couleur sur écran de 14" - entrées RGB et signal de couleurs complet - sélecteur pour utilisation sur écran vert - alimentation auto-régulée par sélecteur - boîtier métallique - conception professionnelle. Existe aussi en vert ou en ambre.

3.100 F

MAGNÉTOPHONE A CASSETTE AVEC CORDON : Branchez-vous sur les prix !
Adaptable grâce à un cordon, il se substitue au micro-drive pour stocker les programmes et permet l'utilisation de toutes les cassettes logicielles.

585 F

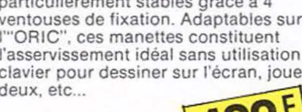
62 F

INTERRUPTEUR : Un seul geste suffit !
Dispositif d'ouverture et de fermeture du contact. Un petit accessoire mais un grand complément de sécurité.



CARTE 8 ENTRÉES - 8 SORTIES : demandez le programme !
Reliée à l'ORIC c'est la fée du logis ! Vous pouvez "enchanter" votre machine à laver ou votre cafetière grâce à cet extraordinaire instrument de programmation et de relais. Une baguette magique signée PERIPH'ORIC.

"JOY STICKS" AVEC INTERFACE : Prenez les commandes de votre ordinateur !
2 poignées de commande inspirées de l'aviation au design aussi élégant que fonctionnel. Agréables, efficaces et particulièrement stables grâce à 4 ventouses de fixation. Adaptables sur l'"ORIC", ces manettes constituent l'asservissement idéal sans utilisation du clavier pour dessiner sur l'écran, jouer à deux, etc...



400 F l'ensemble



Avec cette nouvelle gamme de haut niveau adaptable sur l'ORIC-ATMOS, ORIC fait reculer les limites de l'informatique personnelle.

A la maison ou au bureau, pour la gestion domestique, les jeux ou le travail, vous ferez un bond spectaculaire dans l'espace micro.

Grâce à des prix très étudiés, vous pouvez entrer de plain-pied dans l'informatique totale d'ORIC. Accéder à une technologie de pointe parvenue à son plus haut degré de maturité. Découvrir les applications ergonomiques, ludiques, éducatives infinies de l'informatique personnelle.

La nouvelle gamme PERIPH'ORIC : c'est le moment privilégié d'entrer dans l'informatique totale et définitive d'ORIC.

Alors, qu'attendez-vous ?

LIGHT PEN : dialoguez directement avec votre ordinateur.

Un crayon optique aux performances étonnantes ! Branchez son cordon sur l'ORIC et vous pouvez en un clin d'œil écrire, effacer, corriger, et rajouter à volonté sur l'écran, sans utiliser le clavier.

L'ultime sophistication de la communication informatique personnelle.

450F

MODULATEUR NOIR ET BLANC : Pour exploiter votre ancien téléviseur familial

190F

Muni d'un cordon modulateur, il est indispensable pour relier l'ORIC aux téléviseurs antérieurs à août 79, qui ne disposent pas d'une sortie PERITEL.

MODEM : Entrez aux PTT !

Relié à l'ORIC et à votre téléphone, c'est un système de communication puissant, qui vous ouvre les portes d'une fantastique banque de données : les réseaux télé-informatiques des PTT (système MINITEL). Permet aussi de communiquer avec tous les possesseurs d'ordinateur ORIC.

1.490F

SYNTHÉTISEUR VOCAL : Faites parler votre ordinateur !

Branché sur l'ORIC, il peut parler n'importe quelle langue et son vocabulaire est illimité. Accessible au BASIC. Sortie de contrôle pour haut-parleur à niveau réglable et sortie magnéto pour chaîne HI-FI, ampli, etc...



PERIPH'ORIC

Distribué par ASN, chez votre revendeur agréé ORIC

ASN Diffusion Electronique SA.

- ZI La Haie Griselle BP 48 94470 BOISSY-ST-LEGER
- 20 rue Vitalis 13005 MARSEILLE